2018

Multiple Drone Detection and Acoustic Scene Classification with Deep Learning

Hari Charan Vemula
Wright State University

Multiple Drone Detection and Acoustic Scene Classification with Deep Learning

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

by

Hari Charan Vemula
B.Tech, GITAM University, 2015

2018
Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

12/14/2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Hari Charan Vemula ENTITLED Multiple Drone Detection and Acoustic Scene Classification with Deep Learning BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

_____

John C. Gallagher, Ph.D.
Thesis Director

_____

Mateen M. Rizki, Ph.D.
Chair, Computer Science and Engineering

Committee on Final Examination:

_____

John C. Gallagher, Ph.D.

_____

Mateen M. Rizki, Ph.D.

_____

Thomas Wischgoll, Ph.D.

_____

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Vemula, Hari Charan. M.S., Department of computer science and Engineering, Wright State University, 2018. *Multiple Drone Detection and Acoustic Scene Classification with Deep Learning.*

Classification of environmental scenes and detection of events in one's environment from audio signals enables one to create better-planning agents, intelligent navigation systems, pattern recognition systems, and audio surveillance systems. This thesis will explore the use of Convolutional Neural Networks(CNN'S) with spectrograms and raw audio waveforms as inputs to Deep Neural Networks with hand engineered features extracted from large-scale feature extraction schemes to identify the acoustic scenes and events. The first part focuses on building an audio pattern recognition system capable of detecting the if there are zero, one, or two DJI phantoms in the scene within the range of a stereo microphone. The ability to distinguish the presence multiple UAV's could be used to augment information from other sensors less capable of making such determinations. The second part of the thesis focuses on building an acoustic scene detector to Task 1a in the DCASE2018 challenge(http://dcase.community/challenge2018/index). In both cases, this document will explain the pre-processing techniques, CNN and DNN architectures used, data augmentation methods including the use of Generative Adversarial Networks(GAN's), and performance results compared to existing benchmarks when available. This thesis will conclude with a discussion of how one might expand the techniques in the construction of commercial off the shelf audio scene classifier for multiple UAV detections.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to my advisor Dr. John C.Gallagher for being patient, and guiding me all along the journey of my thesis. I would also, like to thank Google for making the GPU computing accessible to masses for free with Colaboratory.

Dedicated to

My Dad and Mom.

1 Introduction

This thesis work consists of two parts, The first part deals with the problem of multiple drone detection followed by the 2018 Kaggle challenge on Detection and Classification of Acoustic Scenes and Events(DCASE) dataset.

1.1 Drone Detection with audio

1.1.1 Problem Statement

Unmanned aircraft systems, in the last decade has witnessed a significant increase in its usage in commercial, recreational and military applications. The US Federal Aviation Administration[1] defines an unmanned aircraft as a device used for flight in the air with no on-board pilot and whose range in size is from wingspans of six inches to 246 feet, and can weight from approximately forty ounces to over 25,600 pounds. The functional categories of drones include Target and Decoy, Reconnaissance, Combat, Logistics, civil and commercial. The initial applications of unmanned aerial vehicles were limited to the domains of military applications. However, with the increase in the technology, the design and development of the drones have got cheaper, and the domain of their usage has rapidly expanded. Currently, the drones are employed in disaster management[2], product delivery systems[3, 4], search and rescue missions[5], herding a flock of birds approaching airport[6], miniature drones for reconnaissance[7], pesticide delivery in agriculture[8] and

the list is exponentially increasing.

Drones along with their numerous applications in the airspace come with security risks which include airspace threats, privacy, using the vehicle as a weapon, corporate espionage, vehicle collision, and drone-based hacking. There are two phases in designing and enforcing regulations for drone usage, the first phase deals with drone detection and is followed by how to respond in the event if a drone being detected. The scope of this thesis work only focuses on the first phase of drone detection. FAA has implemented ”no-drone-zone” over sensitive areas in the US, which never really worked preventing the drones in appearing in those areas, and a need for a technical solution for detecting drones has called for after the incident of a drone crashing near the white house[9]. One of the most prominent solutions is to force the drone manufacturing companies like DJI to embed a transponder system like GeoSpatial Environment Online lock[10] into the devices to geofence the devices from flying into the ”no-drone-zones.” However, this is not a practical solution, due to the increase in the open technology over the World Wide Web, which enabled drone users to build drone right from scratch, and monitoring them would be a near to impossible task. The military may use very expensive RADAR systems to detect Drones, however, they are expensive, and their design of functionality is not suitable in the urban environment. There also exists a few end-to-end commercial solutions employing wide variety of complex sensory systems like [11] using acoustic sensors, [12] employing radar, radio frequency, acoustic, camera and thermal sensors, [13] employing a radio frequency sensors for RF programmed drones, along with RADAR solid-state Doppler for non-RF programmed drones.

1.1.2 Related Work

As far as the author’s knowledge is concerned, there exists no research in classifying multiple quadrotors in the scene. And, the only second one to employ convolutional neural networks for solving the problem of drone detection. However, few studies were found in detecting the presence of a single quadrotor in the scene[14, 15, 16]. Features like zero-

crossing rate, reflection coefficients, the slope of the spectrum, spectral centroid, spectral roll-off, MFCC's and Mel-spectrograms are used in those studies. [14] in his/her work used Mel-spectrograms with convolutional neural networks, Gaussian Mixture Models and recurrent neural networks, with the reported detection accuracy of 64.15 with CNN and 80.09 with GMM. [16] employed hand engineered features like zero-crossing rate, linear predictive coding representing the spectral envelope of the audio signal, are used with the DSP processor creating the database of the features, during inference the values attained for the corresponding features are compared to the ones in the database in predicting the presence or absence of the drone. In the work of [15], support vector machines are employed on features including short time energy, temporal roll-off, temporal centroid, zero crossing rate, spectral centroid, spectral roll-off, and Mel frequency cepstral coefficients.

1.1.3 Aim and Scope

Unlike naturally occurring sounds, Drones do have distinctive sound characteristics. Leveraging on this, the first part of the thesis work focuses on building an off-the-shelf audio inference system for detection of the multiple drones in the scene. In work here we will use raw spectrograms, log-Melspectrograms, harmonic-percussive source separation and raw audio waveforms of the audio samples collected from stereo microphones (three spectrograms per sample, one for each channel of the stereo input and one for the difference between the two channels) as an input to the CNN's to determine the number of drones present in the area. The use of spectrograms and raw audio waveforms as inputs converts the detection problem into a vision problem. The natural positional scaling abilities of the CNN capturing spectro-temporal features should come into play.

This part of the thesis seeks to answer the following questions:

1) Is it possible to build an audio inference system for detecting the presence of multiple drones in the area with inexpensive Commercial Off the Shelf Equipment(COTS)?

2) Assuming it can work in the prototype form, what challenges might one face in scaling

drone detection for practical use?

3) Could the techniques used in the drone detection system prototype beat the commercial drone detection systems in performance?

The questions will be addressed in the context of a comprehensive comparison between the performance of the DNN with large-scale feature extraction schemes, CNN with variants of spectrograms in both frequency scale and the psychoacoustic scales and the performance of the recent sample level CNN's architectures for the detector working on raw audio forms in time-domain.

1.2 Detection and Classification of Environmental Sound Events

1.2.1 Problem Statement

Classification of Environmental sound events is a subfield of computational auditory scene analysis which focuses on the creation of intelligent machine listening systems identifying acoustic scenes similar to human listeners. The applications include tagging millions of hours of audio on the web, increased precision in understanding the environment for autonomous systems, audio surveillance systems, noise mitigation and off-the-shelf pattern recognition systems. Since the year 2010, it has been observed how benchmark datasets like Imagenet paved the way for the field of computer vision to achieve above human-level performance on Image classification tasks. The DCASE data set is the Imagenet of Acoustic scene and event classification. To further expand the intuition developed in part 1, this part of the thesis focuses on the classification of environmental sound events. A baseline system employing a two layer convolutional neural networks with a reported accuracy of

59 percent is provided by the challenge organizers.

1.2.2 Related Work

Since the inception of the DCASE challenge in the year 2013, over 50 submissions are made every year, However, until 2016 submissions were based on statistical and probabilistic models with hand engineered features[17, 18, 19, 20, 21, 22]. Since 2016, The convolutional neural networks have been employed in one form or the other in almost all the state of the art approaches[23, 24, 25, 26, 27, 28].

1.2.3 Aim and Scope

Unlike, the sounds obtained in Automatic Speech Recognition (ASR) and environmental event detection the acoustic scene detection sounds are continuous. In the detection of an acoustic scene class, firstly, all the events happening in the scene are identified, and by learning the relationship between the detected events will help in predicting the acoustic scene class. It becomes harder in hand engineering the features which capture such complex relationships. In the scope of this thesis work, large-scale feature extraction schemes were employed followed by spectrograms and its variants including Mel Spectrograms in log scale.

The final objective is to attain far off better performance over the baseline system, Understanding the suitable CNN architectures in solving the problem.

This part of the thesis seeks to answer the following questions:

1) How would the performance of the deep learning models working on raw data in the spectro-temporal domain, compare to the performance of the statistical and probabilistic models working with Hand engineered features of both time and frequency domain?, What is the effect of the infinitely strong prior of CNN's over its weights on the solution?

2) Could the field of Environmental Sound classification leverage the algorithmic advances

in the field of computer vision and deep learning. Which include the effect of Batch Normalization, Exponential Linear Units(ELU), Rectified Linear Unit (ReLU), addition of Gaussian Noise in training, and Cyclic Learning Rates(CLR).

3) Which of the CNN architectures yield better generalization performance and faster convergence with spectro-temporal data.

4) Which of the variants of spectrograms can yield better performance with Convolutional Neural Networks.

5) What is the performance of convolutional Neural Networks on raw audio waveforms in time-domain?

1.3 Document Overview

This document is organized as follows: This document is organized as follows: Chapter two will provide the background information on the techniques used for the Signal Processing, Feature Extraction and Classification algorithms employed in the scope of this thesis work along with the in brief discussion on the openSMILE large-scale feature extraction library. Also, it discusses the data collection environment and the dataset organization for the custom collected dataset for multiple drone detection. Chapter three discusses the performance metrics used in the scope of this thesis work. Chapter four and Chapter six deals with the experiments performed on the multiple drone detection dataset, and Chapter seven deals with experiments performed on the DCASE dataset. Chapter five explains the process involved in augmenting the drone detection dataset. Chapter eight discusses the results obtained for both datasets, along with the comparison between the performance of architectures and feature extraction schemes employed in the experiments. Followed by it discusses the comparison with the objective of this thesis with the results obtained and the significant observation that was made in the way through the completion of this thesis, and the future work and the possible extension that could be made to this work.

2 Background

This chapter provides the background information needed for the design and deployment of the multiple Drone detection system and the Acoustic Scene classifier. The pipeline components involved in building a Drone detection system and Acoustic Scene classifier are as follows The pipeline to build an audio-based pattern recognition system involves the following steps.

1) Dataset collection.

2) Dataset Preprocessing

3) Feature Extraction

4) Model Training

5) Deploying the best performing classifier for inference.

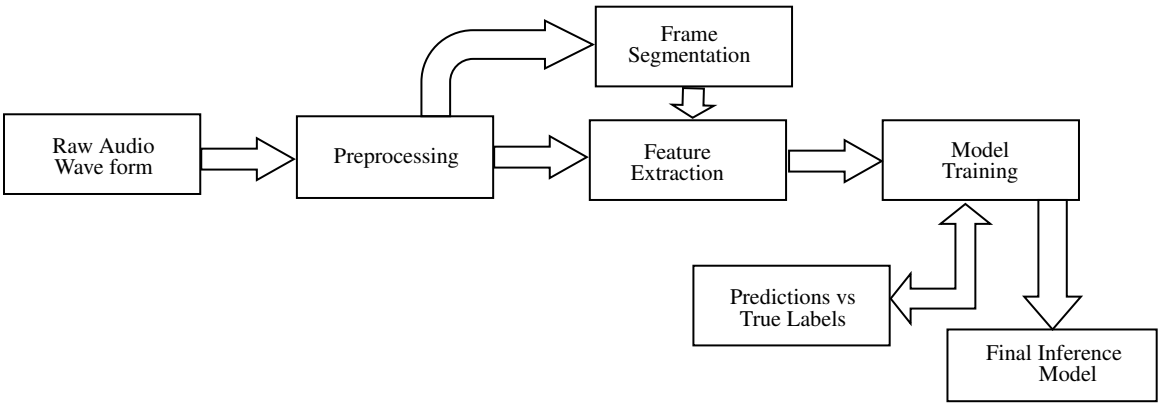


Figure 2.1: Pipeline Diagram for Design of Drone Detection system

2.1 Dataset Design

2.1.1 Drone Detection Dataset

Due to the limited research available in the field of drone detection with audio analysis, there are no public datasets available. Hence, a custom dataset was collected for three classes of multiple drone detection which include background noise, a single drone in the scene and two drones in the scene. The architecture of the dataset makes it a multi-class classification problem. The audio for DJI phantom was recorded at 44100Hz, CD-quality sampling rate with 16bit resolution and two audio channels. The recorded audio was stored on the disk as uncompressed WAVE files. The recordings are collected in 3 different locations to introduce acoustic variability in the dataset. The recording locations include a laboratory, a hallway, and an emergency exit staircase. In each scene, a single recording was made for the classes background noise and two drones in the scene. For the class, Single Drone in scene 4 recordings is made in the scene lab and two recordings in the scenes hallway and staircase.

Hardware

The hardware employed in dataset collection include a Sony ECM-DS70p-portable stereo microphone and two DJI phantom standard 3's.

Dataset Setup

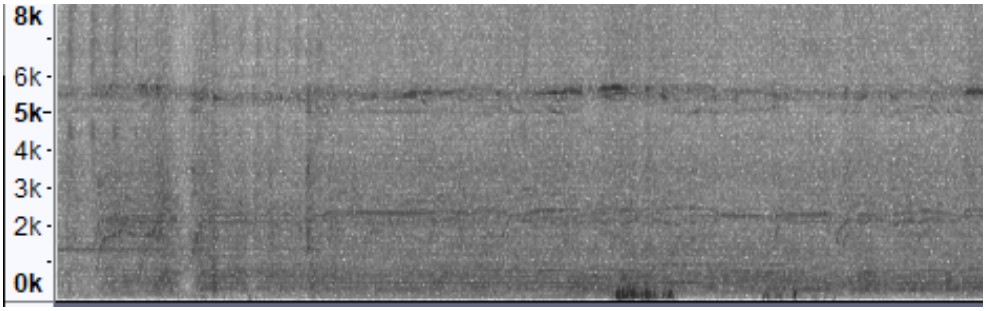
Each recording from the drone detection dataset is later split into 100 samples with a duration of one second for each sample, resulting in a dataset of size 1400 samples. The dataset is unbalanced with 300 samples each for classes background noise and two drones in the scene, followed by 800 samples in class single drone in the scene. Three-fold cross-validation is employed by placing the audio samples from each scene into a fold. The

(a) Sony ECM-DS70p-portable

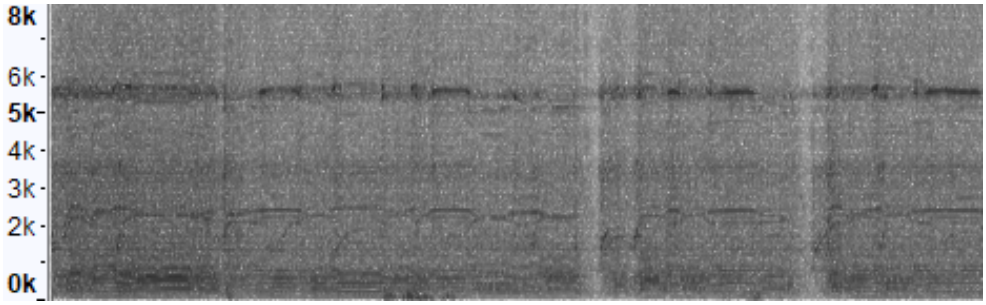
(b) DJI Phantom Standard 3

Figure 2.2: Hardware used in Recording Drone audio

anatomy of the Dataset is found in the table 2.1.



(a) Spectrogram of Single Drone in the scene



(b) Spectrogram for two Drones in the scene

Figure 2.3: Sample spectrograms Drones audio

Location	Background Noise	Single Drone in Area	Two Drones in Area
Lab	100 seconds	400 seconds	100 seconds
Hall Way	100 seconds	200 seconds	100 seconds
Emergency Exit Staircase	100 seconds	200 seconds	100 seconds

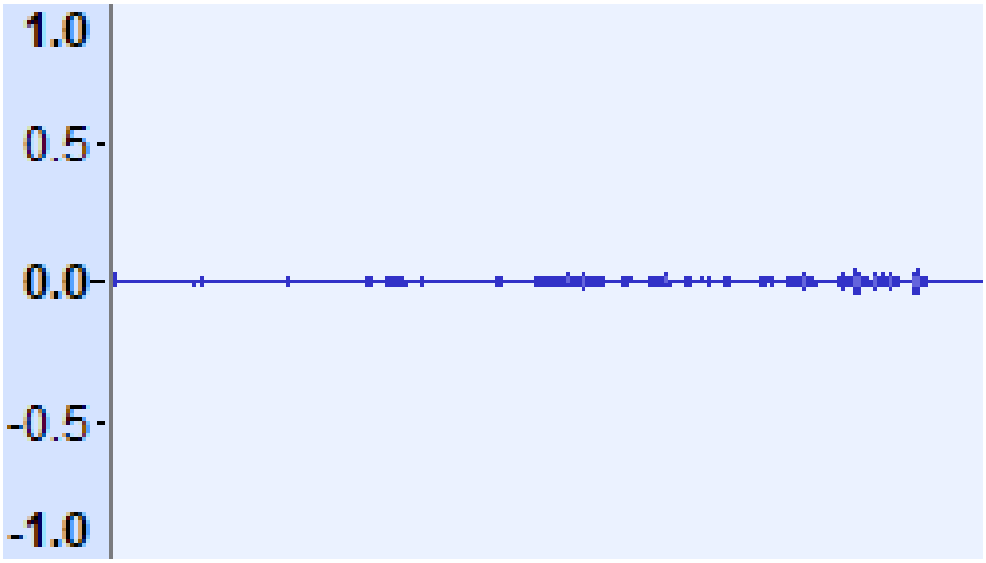
Table 2.1: Anatomy of Drone detection dataset

Amplifying the Audio Signals

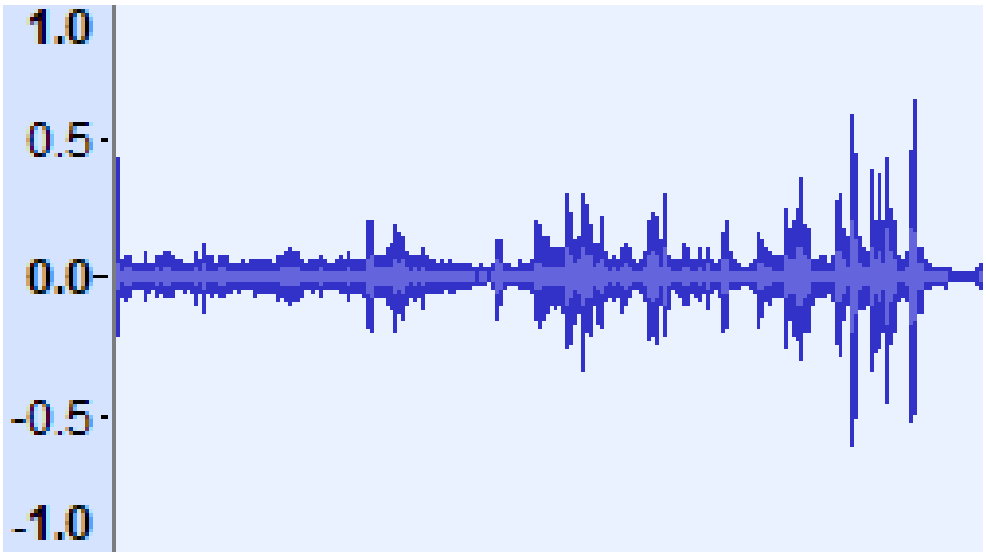
The audio samples in the drone detection dataset do not have enough amplitude for human hearing. To evaluate the human performance on the classification task, each audio sample in the drone detection dataset is amplified. The raw audio waveforms for the class single drone in the scene before and after performing the amplification is shown in the figure 2.4.

2.1.2 DCASE Dataset

The dataset consists of 10 Acoustic scene classes recorded in six different locations. The original recordings were made for 5-6 minutes in each location, with a sampling rate of 48000 Hz and 24-bit resolution recorded with Soundman OKM II Klassik/studio A3, electret binaural microphone and a Zoom F8 audio. To make the recorded audio similar to the sound reaching the human auditory system, the microphones are shaped in the form of headphones and are worn around the ears. The recordings were later split into individual files of 10-sec duration resulting in the current dataset with 8640 files for ten classes. The training and validation split is performed such that for each class there exists no overlap between the recording locations resulting in 6122 WAVE files in the training set and 2518 files in the validation set



(a) Audio sample for Single Drone in the Lab scene



(b) Amplified signal for Single Drone in Lab scene

Figure 2.4: Signal Amplification

Acoustic Scenes	Num Samples Training	Num Samples Validation	Recording Locations
Airport	599	261	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Bus	622	242	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Metro	603	261	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Metro Station	605	259	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Park	622	242	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Public Square	648	216	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Shopping Mall	585	279	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Street Pedestrian	617	247	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Street Traffic	618	246	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Tram	603	261	Vienna, London, Helsinki, Stockholm, Barcelona, Paris
Total	6122	2518	Vienna, London, Helsinki, Stockholm, Barcelona, Paris

Table 2.2: Anatomy of the DCASE dataset

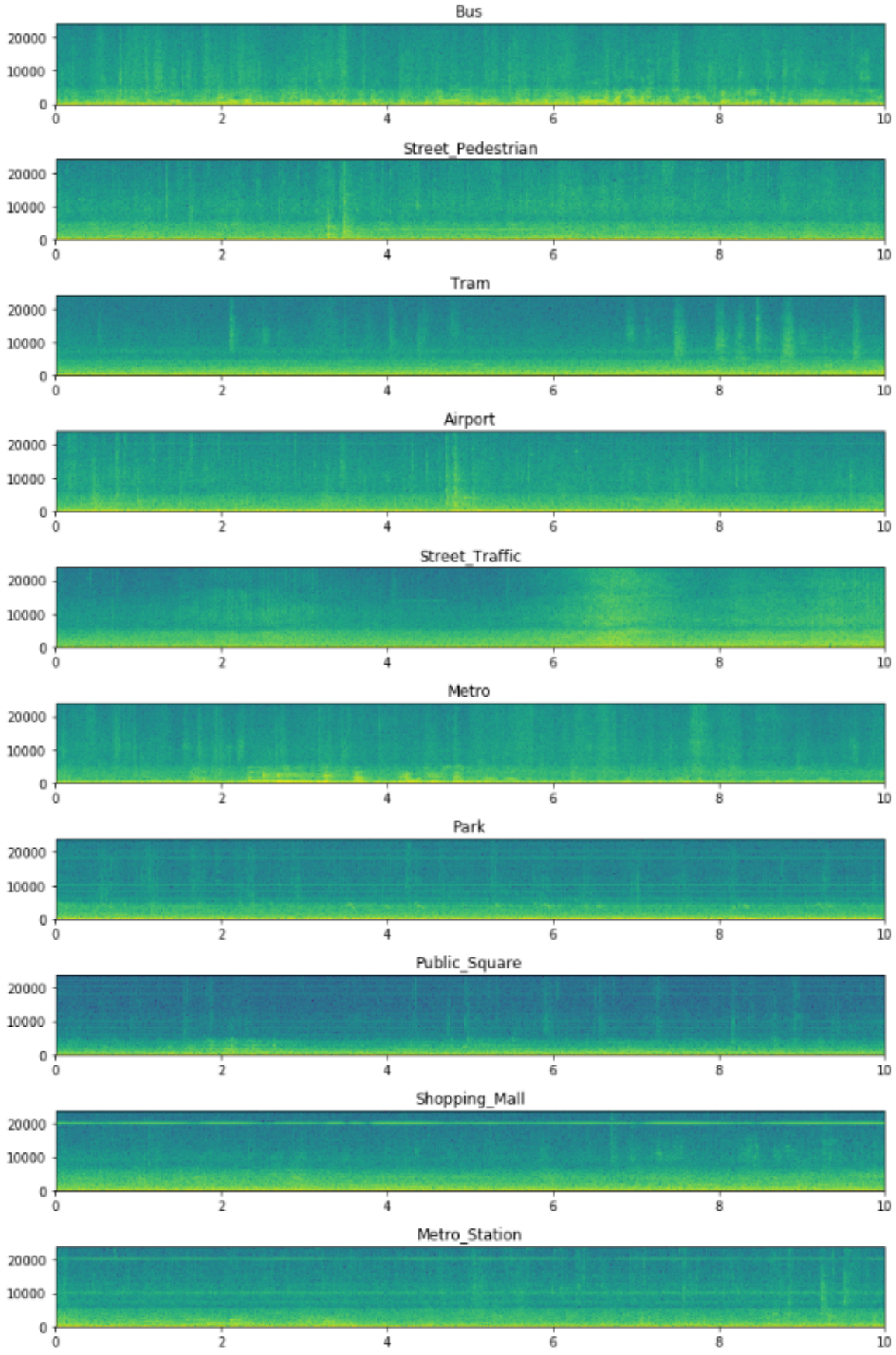


Figure 2.5: Sample Spectrograms for DCASE classes

2.2 Mathematical View of Signal Processing

2.2.1 Nyquist-Shannon Sampling Theorem

The Nyquist-Shannon sampling theorem[29], provides a condition for converting an analog signal into evenly spaced discrete samples, such that the reconstruction of analog signal from a discrete signal is possible. Also, it eliminates the effect of aliasing. Aliasing makes multiple signals indistinguishable from each other.

For a band limited signal, whose Fourier transform is non-zero only for certain frequencies, If f_{max} is the maximum frequency component in the analog signal, the condition for sampling theorem states that the sampling frequency should be greater than twice the maximum frequency component.

$$F_s > 2f_{max} \tag{2.1}$$

2.2.2 Convolution Theorem

Convolution theorem[30] states that the Fourier transform of convolution of two signals is equal to the point wise product of their Fourier Transforms. It is also interpreted as the convolution operation in the time domain is equal to the point wise multiplication of the signals in the frequency domain. In the scope of this thesis, this provides the foundation for using the CNN with raw audio waveforms with the first convolution blocks of the networks approximating the filter to perform the Fourier Transform.The equation that describe the convolution theorem are as follows.

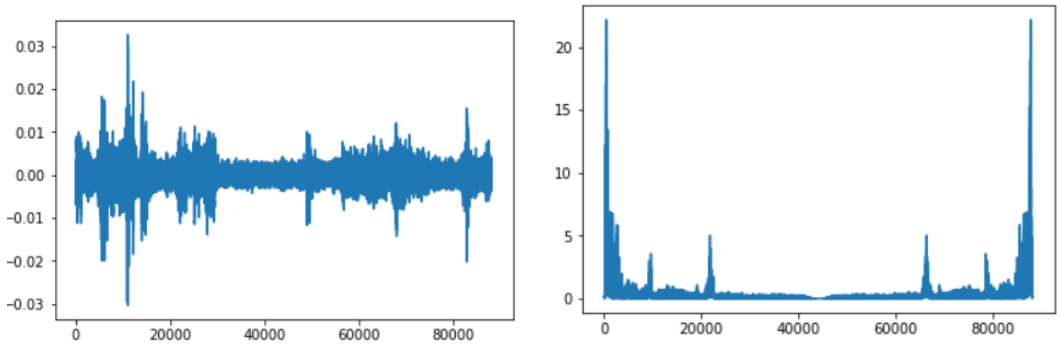
$$\mathcal{F}(f * g) = \mathcal{F}(f).\mathcal{F}(g) \tag{2.2}$$

2.2.3 Discrete Fourier Transform

All the signals observed in the nature can be decomposed into sum of pure sinusoids with different frequencies. Fourier Transform is a mathematical technique for obtaining the spectral composition of the signal by decomposing a signal into pure frequencies that make up the original signal. The resulting sinusoids of Fourier Transform on a signal represented as a function of time is a complex value, whose imaginary part represents the phase offset of the pure sinusoid and and its absolute value represents value of the corresponding frequency component. Applying inverse Fourier Transform on the resulting signal reconstructs the original signal when the condition provided for sampling theorem is satisfied. The Fourier Transform applied on discrete signals is called Discrete Fourier Transform[31]. The limitations of the Discrete Fourier transform were it cannot yield representations for time variant, non-stationary signals.

The mathematical equation of the DFT is:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\Pi i k n/N} \tag{2.3}$$

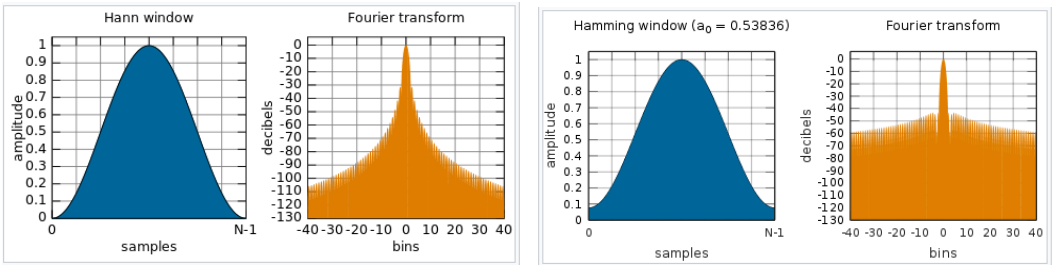


(a) Raw audio waveform (b) Frequency domain signal after applying DFT

Figure 2.6: Discre Fourier Transform.

2.2.4 Short Time Fourier Transform

Applying Fourier Transform on the signal changes its representation from Time-domain to Frequency-domain with the loss of the temporal information. STFT[32] algorithm provides a way to perform the Fourier Transform without loosing the temporal information. In STFT, the signal is decomposed into overlapping frames employing an window function to smooth out the irregularities at the edges of frames. The resulted frequency domain representation of individual frames are stacked together in the frequency axis resulting in a spectro-temporal representation capturing both the time domain and frequency domain information. An ω[n] is the analysis window applied to the signal, The variants of existing window functions are the Rectangular window, Triangular window, Hann, and Hamming windows. In the scope of this thesis hamming and Hann windows are used.



(a) Hann Window (b) Hamming Window

HTTP://en.wikipedia.org/wiki/Window_function

Figure 2.7: Window functions

with x[n] representing the signal and ω[n] representing the window the equation for STFT is:

$$X(n,\omega) = \sum_{m=-\infty}^{\infty} x[m]e^{j-2\Pi kn/N}w[n-m] \tag{2.4}$$

Taking into account the Heisenberg uncertainty principle for signal processing[33], the perfect time-frequency representation signal could never be known. A better frequency resolution results worsens the time resolution of the signal and vice-versa.

The equation for Heisenberg uncertainty principle in signal processing is

$$\Delta t * \Delta f \geq \frac{1}{4\pi} \tag{2.5}$$

Narrowband Spectrograms are obtained with longer analysis window exhibiting harmonic structure and precise location of transitions, whereas wideband spectrogram is attained with short analysis window showing the temporal structure and high-frequency resolution with no ability to localize frequency domain.

2.2.5 Mel-Scale

Converting frequency domain to mels domain is done using formula:

$$m = 2595 log_{10}(1 + (f/700)) \tag{2.6}$$

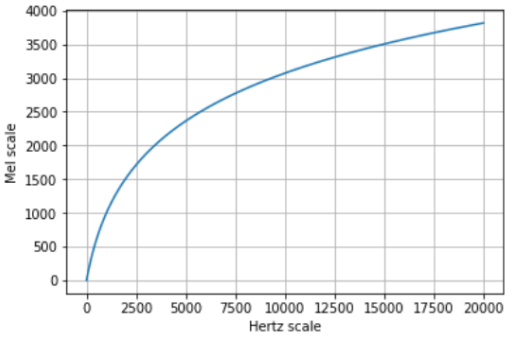


Figure 2.8: Mel scale vs Hertz scale.

Mel-Scale is the psycho acoustic representation of frequency in linear scale. The cochlea present in the Human auditory system acts as a critical bandpass filter, each of the membranes present in cochlea vibrates to the certain frequency component. To imitate these properties in audio processing,Stevens, Volkmann, and Newmann in 1937 proposed an unit of pitch called 'Mel'. 'Mel' is defined as the perceptual scale of pitches judged by listeners to be equal distance from each other. During the series of experiments, it was

observed that when the frequency of the signal is less than 1000Hz, human auditory system perceive signal on a linear scale and for the frequency, over 1000Hz it was recognized on a logarithmic scale. The essence of Mel-scale is to bring this feature into perspective.

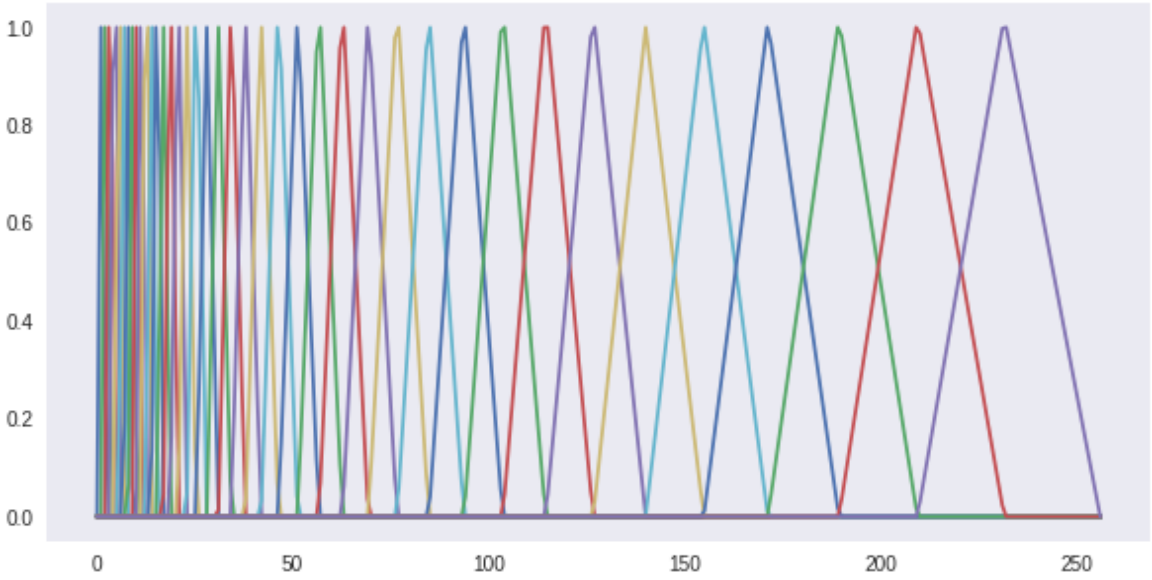


Figure 2.9: Mel Filter Bank with 40 mel filters

2.2.6 Log-MelSpectrogram

A series of Triangular Mel filters are applied on the result of the STFT on raw data such that more filters are used on low-frequency regions, and less number of filters are employed on the high-frequency regions of the power spectrogram. This filter's behavior imitates the auditory filters that are present in the human ear. Mel filters allow the power spectrogram to be mapped on to Mel-Scale, applying logarithmic transformation at each Mel frequencies result in Log-Mel spectrogram. To date, Log-MelSpectrograms is considered as one of the best variants of the visual features that could be used as an input feature to convolutional neural networks, specially for the tasks of Automatic Speech Recognition(ASR) and Sound Event Detection(SED)

2.2.7 Discrete Cosine Transform

DCT is a mathematical technique applied to the log-melspectrogram resulting in Mel frequency cepstral coefficients. The operation of DCT is similar to DFT, and The critical difference is unlike DFT, DCT consists of only cosine terms which are real.

The equation of the DCT is:

$$X_k = \sum_{n=0}^{N-1} x_n cos(\frac{\pi k(2n+1)}{2N}) \tag{2.7}$$

2.2.8 Harmonic Percussive Source Separation

Real life audio consists a mixture of signals with different frequencies, and if there exists a signal which is an integral multiple of another reference signal in the same audio is called harmonic component. While a percussive component of the signal is defined as the part of audio generated when two objects strike each other. In most of the cases, the harmonic component of the sound is observed in the horizontal direction, while percussive components are seen in the vertical direction of the spectro-temporal representation. In simplest terms, harmonic sounds have pitch, while percussive sounds have perfect localization in time.

Techniques like median filtering [34], Nonnegative Matrix Factorization[35, 36] are used for separating the harmonic and percussive components from the audio. These algorithms work with the spectro-temporal representation of the audio, after the separation of the harmonic and percussive components, Inverse Fourier Transform(IFFT) can be applied on both the components to obtain the time-domain representation of the respective components.

Figure 2.10 shows the spectrograms of the original signal, followed by harmonic and percussive components.

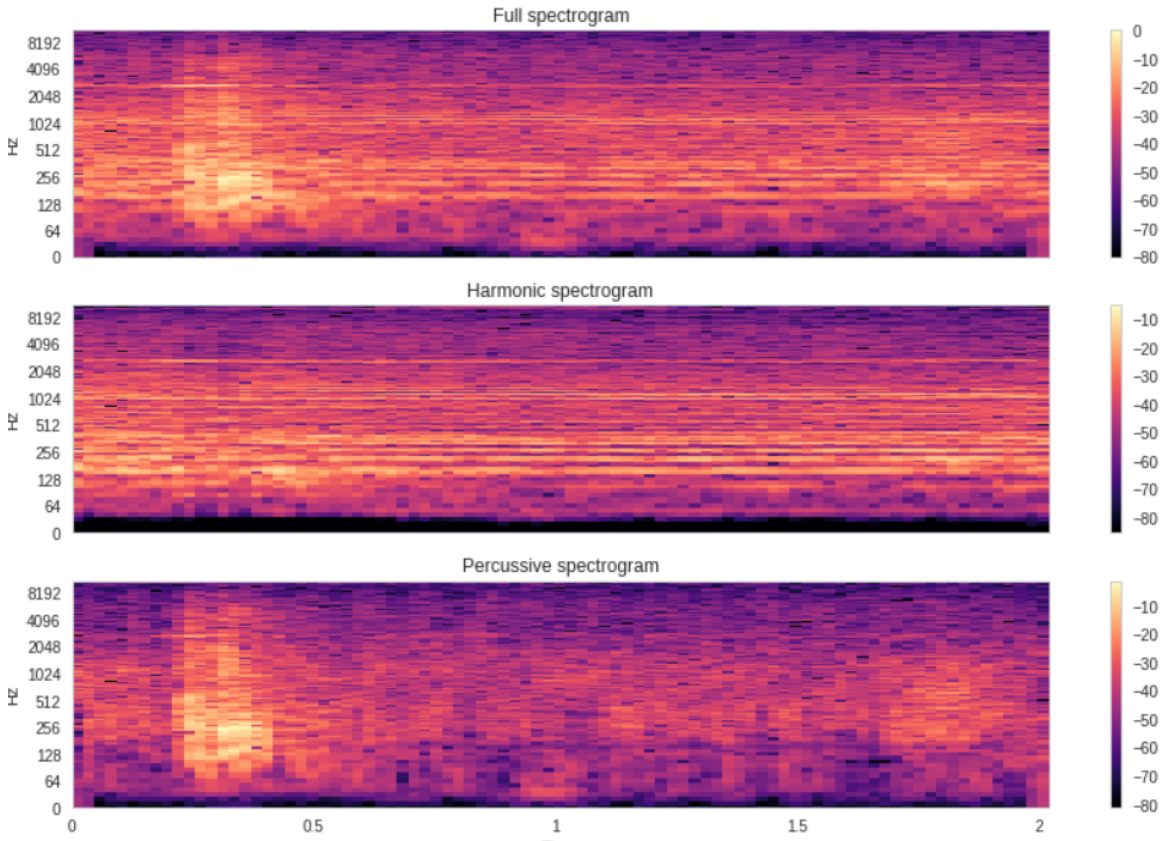


Figure 2.10: Harmonic Percussive Source Separation

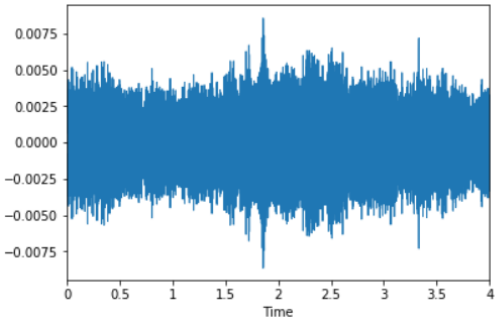
2.3 Comparison between variants of Spectrograms

The main disadvantage of employing an STFT algorithm is, once a window size is chosen, the same time-frequency resolution exhibited on the entire range of the audio signal. However, the real-world audio displays a diverse range of frequencies. Also, it is impossible to choose the size of the window, until the signal is manually analyzed. However, the Mel-scaled spectrograms overcome these limitations.

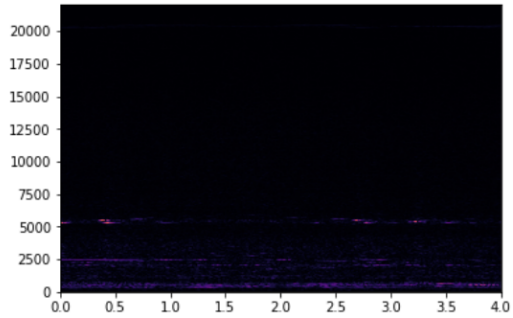
The Mel-scaled spectrograms are designed focusing on imitating the human auditory system, by placing the higher number of filters on the lower end of the frequency and lower number of filters on the higher end of the spectrum.

Due to the nature of its design, Mel-spectrograms are well suited for the tasks of Automatic Speech Recognition(ASR). However, raw spectrograms are more robust to noise compared to the Mel-Spectrograms, which makes it a greedy representation when working with the tasks of multiple Drone detection and Acoustic Scene Classification(ASC).

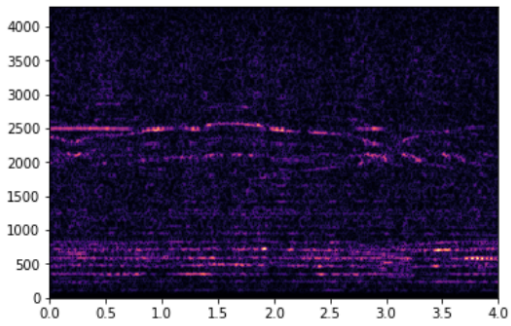
Figure 2.11 shows various descriptions of spectrograms employed in the scope of this thesis work.



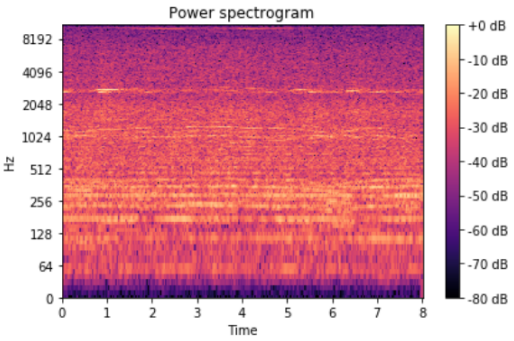
(a) Raw audio waveform



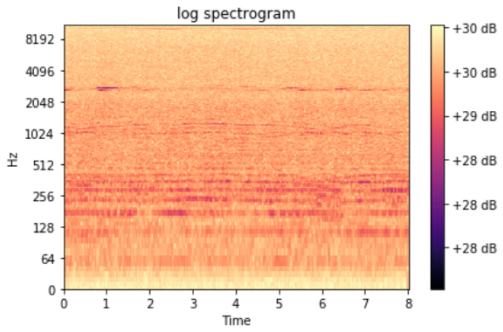
(b) Spectrogram



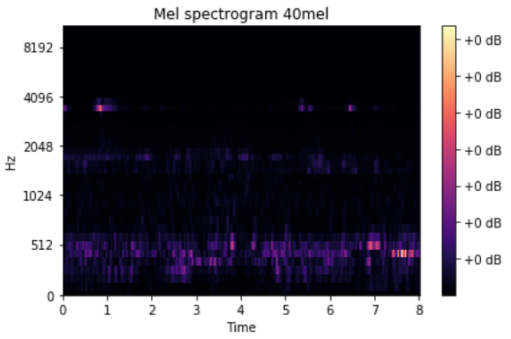
(c) Zoomed Spectrogram 200 bins



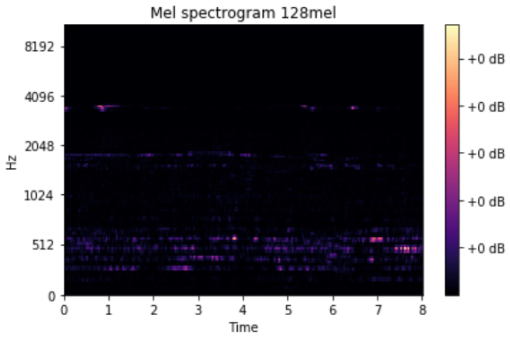
(d) power spectrogram



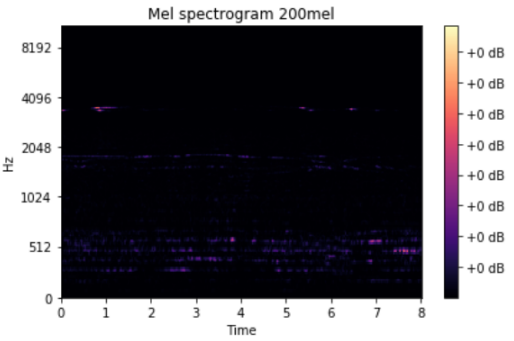
(e) log Spectrogram



(f) melSpectrogram 40 mels



(g) melSpectrogram 128 mels



(h) melSpectrogram 200 mels

Figure 2.11: Visual features of sound in spectro-temporal domain.

2.4 Convolutional neural networks

CNN belongs to the class of feedforward neural networks, which are optimally suitable for unstructured data which has a grid-like topology[37]. The distinguishable features of CNN compared to the DNN, and RNN includes, sparse connectivity, local connectivity, and shareable parameters. These properties contribute to the reduction in time and space complexity of the network. The same set of parameters are shared across the entire input to the convolutional layer, where the activations of the following layer are resulted due to the connectivity of the neuron to local regions in the previous layer. The shareable weights enable the network to learn the equivariant representations of the input.

The enables CNN to learn the hierarchical representation of the input data, with the first layers learning the simple representations like edges and borders, followed by the layers learning the complex relationships in the data with the representation learned by first layers. The important property of the CNN is translational invariance which makes the Network robust to small changes in the input, which is achieved with the presence of the Pooling layers in the network.

Since [38], the CNN's have been dominating the field of computer vision. This continuous progress has resulted in several types of architecture's for CNN which include, VGG[39]), ResNet[40] and Inception Networks[41]. The advent of the regularization algorithms like Batchnormalization[42], Dropout[43] and optimization algorithms including adam[44], and RmsProp has enabled to train deeper architectures with millions of parameters. The sample architecture of the convolutional neural networks is shown in the figure 2.12.

In general, all the convolutional neural network architectures consists of the following layers.

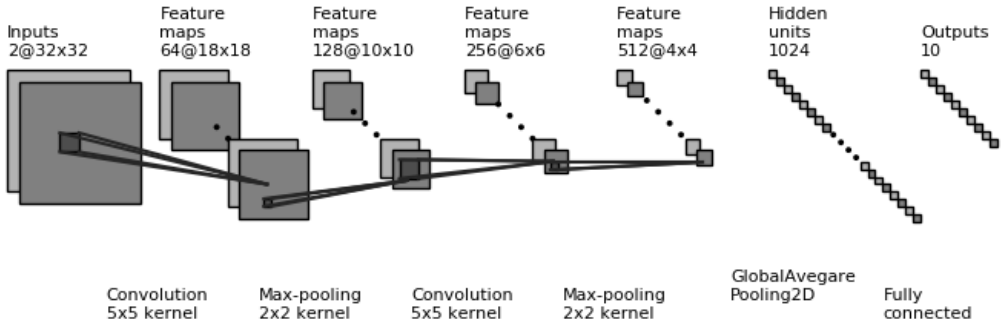


Figure 2.12: Two channel Convolutional Neural Network Architecture for audio classification

2.4.1 Convolutional layer

Convolutional layer, in short, Conv layer consists of a set of four-dimensional filters. The first and second dimension of the filter corresponds to height and width of the filter, followed by the third dimension representing the number of channels of the input, and the fourth dimension corresponds to the number of filters employed. Conv layer performs a mathematical operation called convolution, hence its name, between the input tensor and the set of filters. The convolution operation involves flipping the filters by 180°and summing the result of element-wise product between the flipped filters and the input tensor. However, unlike the fields of mathematics and signal processing, the convolution operation performed in the scope of CNN does not involve kernel flipping, and strictly speaking, this operation is called cross-correlation. However, in CNN literature it is interpreted as convolution operation since the filters are capable of learning weights when filters are not flipped relative to the case where the filters are flipped. The sample convolution operation performed on single-channel input and a single channel filter is shown in figure 2.13. The filters in CNN are also called kernels. The hyper parameters that control the shape of the output volume of the layers in the CNN are padding, stride and depth.

Stride

Stride is represented by a integral number which interprets the amount of shift the kernel is eligible to make across both the height and width dimension of the input in calculating the output feature map of the convolutional layer.

Padding

Padding is defined as the addition of the dummy pixels around the input of a convolutional layers, usually to preserve the dimensions of the input tensor corresponding to the output feature map. The common forms of padding are zero padding and the reflection padding.

Depth

The depth of the output volume of a convolution layer is defined by the number of filters applied on the input tensor.

Let $n_{H_{prev}},\ n_{W_{prev}}\ and\ n_C$ be the height, width and the depth of the input image, $f,\ f,\ n_{prev},\ n_C$ be the height, width, the depth of the input, which defines the shape of the filter and $_f$is the number of filters applied. With n_H, n_W and n_C as the dimensions of the output volume of a convolutional layer with a stride 's' and padding 'p' the output of the Convolutional layer is given by

$$n_H = [\frac{n_{H_{prev}} - f + 2 * p}{s}] + 1 \tag{2.8}$$

$$n_W = [\frac{n_{W_{prev}} - f + 2 * p}{s}] + 1 \tag{2.9}$$

$$n_C = number\ of\ filters\ applied \tag{2.10}$$

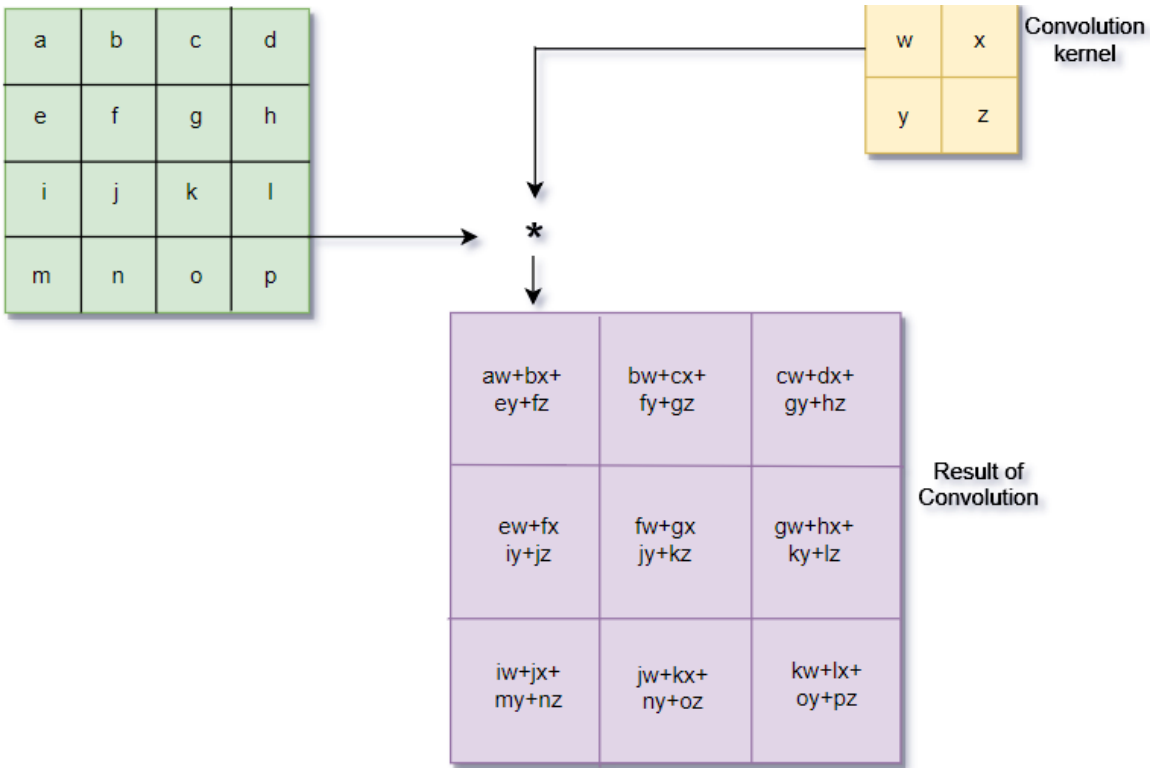


Figure 2.13: Convolution operation in CNN

2.4.2 Activation Layer

The activation layers employees activation function over individual entities of a feature map. This transforms the input from linear hypothesis space to non-linear hypothesis space. Commonly used activation functions include Rectified Linear Units(ReLU), Exponential Linear Units(ELU), sigmoid and Softmax. The dimensions of the input and output tensor of an activation layer are the same.

2.4.3 Pooling Layer

Along with the activation layer, Pooling layer is a non-parametric layer in the CNN; it is employed to reduce the dimensions of the input tensor and to make the network insensitive to the small changes in the data due to noise. The most common types of Pooling operations are Maximum Pooling which yields the maximum value in the analysis window to the

output tensor, and Average Pooling which yields the average of all individual values in the analysis window to the output tensor. The presence of the Pooling layers is responsible for the CNN's property of translational invariance. The sample pooling operation performed with a pool size of 2 and a stride of 2 is shown in figure 2.14.

Let n_{H_prev}, n_{W_prev} and n_{C_prev} be the height, width and the depth of the input to the Pooling layer, with stride s the output volume of the Pooling is defined by the following.

$$n_H = [\frac{n_{H_{prev}} - f}{s}] + 1 \tag{2.11}$$

$$n_W = [\frac{n_{W_{prev}} - f}{s}] + 1 \tag{2.12}$$

$$n_C = n_{C_{prev}} \tag{2.13}$$

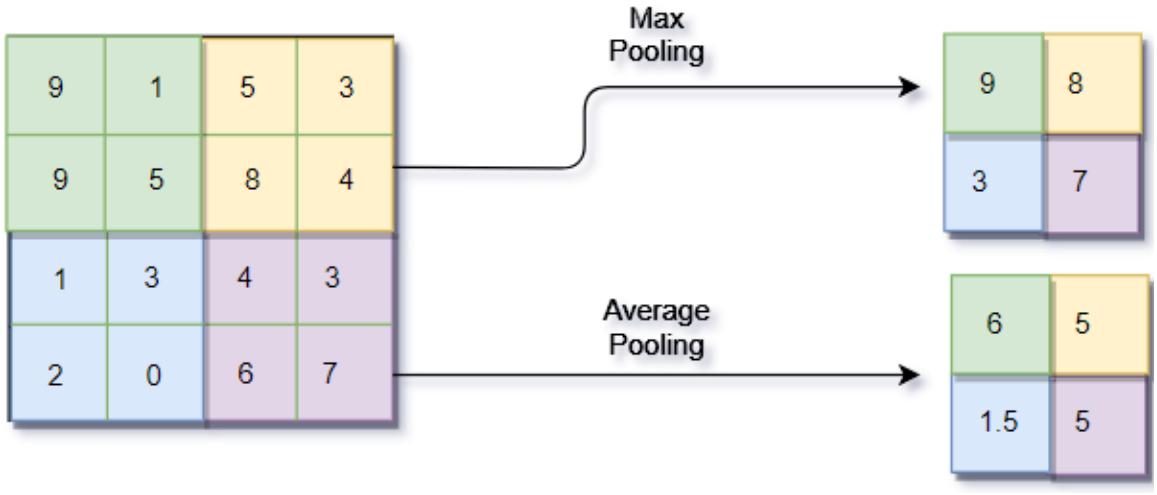


Figure 2.14: Max and Average Pooling.

2.4.4 Dense Layers

Dense layers are employed in the last layers of the CNN, which are similar to the fully connected Feed Forward Neural Network. The convolutional and pooling layers act identically to the sensory organs of the human body generating a hierarchical representation of input features, which are later fed into the Dense layers. The Dense layers take this representation as the input and act like a regular ANN classifier with no shareable parameters and dense connections. The last Dense layer of the CNN outputs the class probabilities.

The back-propagation of error gradients in convolutional neural networks are found in [45]

2.5 Hyper Parameters

The model parameters defined by the user before the beginning of the training, which is capable of determining the capacity and the complexity of the model are defined as hyper-parameters. And once set, their values remain constant. The process of choosing the values for these parameters to increase the performance of the model over a series of training loops is called hyper-parameter tuning. In the recent decade, several algorithmic and regularization techniques were proposed, which resulted in a significant increase in the performance of the models. The hyper-parameters employed in the scope of this thesis work are as follows.

2.5.1 Gradient Descent:

The core idea of training a supervised learning model is to approximate a function which maps the input feature vector to the labeled output, which can later be used for inference. Gradient Descent is defined as an algorithmic technique employed to update the parameters of a function such that it minimizes the difference between the function output and desired

output. It is achieved by iteratively moving towards the lowest point on the error surface by the direction provided by the negative of the gradient.

With $J(\theta)$ as the hypothesis function, α as the learning rate, and $\nabla_\theta J(\theta)$ as the gradient of the hypothesis function three variants of implementation schemes exists for Gradient Descent depending on the number of samples used to update the function parameters, which are described in the following subsections.

Stochastic Gradient Descent(SGD)

In SGD the error is calculated after performing the forward propagation of each sample, and the function parameters are updated. The advantages of SGD include faster computation for more massive datasets and faster convergence since the function parameters are frequently updated. However, frequent updates result in high noise during training and high variance[46] in the gradient. The mathematical equation to perform SGD is as follows

$$\theta := \theta - \alpha.\nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.14}$$

Batch Gradient Descent

In Batch Gradient Descent the function parameters are updated only after calculating the error for the entire dataset. It is computationally faster with the datasets of reasonable size, has a stable convergence with minimal variance in gradient. However, usually, it doesn't reach the optimal convergence and keeps oscillating around the optimal convergence point. The mathematical equation to perform Batch Gradient Descent is as follows.

$$\theta := \theta - \alpha.\nabla_\theta J(\theta) \tag{2.15}$$

Mini-Batch Gradient Descent

It is the most used variant of the Gradient Descent algorithm, which leverages the advantages of both Stochastic and Mini-Batch Gradient Descent. In this, the function parameters are updated after calculating the error for 'N' number of samples usually called mini-batch. Here, the mini-batch size 'N' is also a hyper-parameter, which is chosen by the user. The mathematical equation to perform Mini-Batch Gradient Descent is as follows.

$$\theta := \theta - \alpha.\nabla_\theta J(\theta; x^{(i:i+N)}; y^{(i:i+N)}) \tag{2.16}$$

2.5.2 Optimizing Gradient Descent

The recent research in Deep Learning also resulted in algorithmic techniques, to further optimize the gradient descent. The optimizers used in the scope of this thesis work are shown in the following paragraphs.

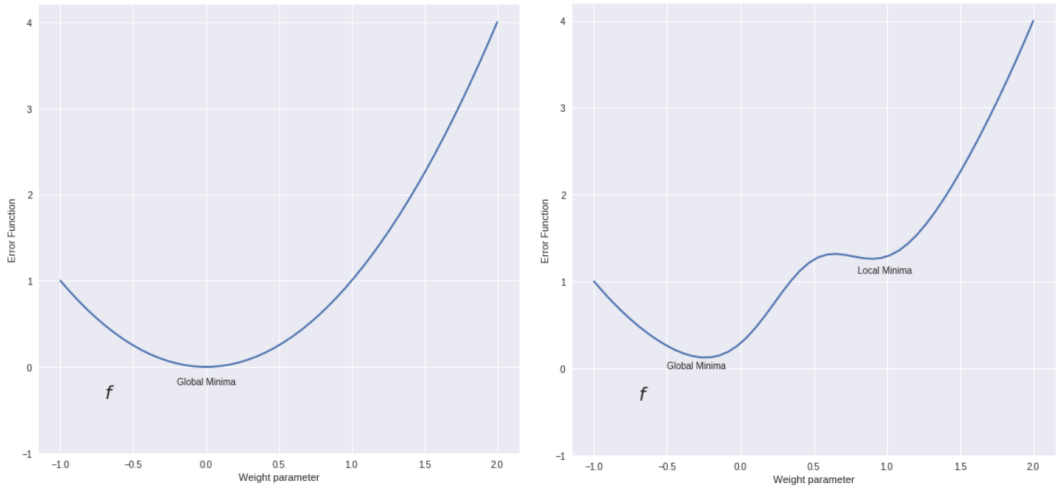
Learning Rate It is the hyper-parameter with the highest precedence in training the supervised learning neural network models. It is denoted by α. The negative gradient obtained on the hypothesis function is scaled by the learning rate and is added to the parameter as an update. Higher values of learning rate rise the problem of overshooting the optimal point or global minimum in the error surface of the hypothesis function, while the lower values of the learning rates results are convergence time by many folds. There exists no right value for choosing the correct value for the learning rate parameter and should be selected by examining the performance of the model by varying it. Several techniques like cyclic learning rates [47], stochastic gradient descent with warm restarts[48], and differential learning rates have shown a significant increase in performance of the neural networks.

Momentum: The real world error functions are not perfectly convex with a single global minima[49], as shown in figure 2.15(b), one or more local minima exist in complex er-

ror surfaces. When naive SGD approaches ravines caused by local minima, it takes a very wavering update increasing the convergence time. Momentum is a term added during parameter update which enables us to increases the size of the steps taken in parameter updates when the gradient points in the same direction decreasing the convergence time, also, yields smoother variations in the case when the gradient is changing its direction frequently. Momentum also enables the error function to jump over the local minima and reach global minimum. With v_t and v_{t-1} as the gradient of the current and the previous step respectively, and β as the momentum the equations for Gradient Descent with momentum are as follows.

$$v_t = \beta_1 v_{t-1} d\theta + (1 - \beta_1) d\theta \tag{2.17}$$

$$\theta := \theta - \alpha . v_t \tag{2.18}$$



(a) Ideal Convex Error Surface (b) Real Convex Error Surface

Figure 2.15: Convex Error Surface

RMSprop: [50] defines RMSprop as a method to accelerate the mini-batch training by, Dividing the learning rate for weight by a running average of the magnitudes of recent

gradients for that weight. With s_t and s_{t-1} as the exponential average of the squares of the gradients, in current and previous time step and β as the decay parameter whose value is usually chosen to be 0.9, the equations for RMSprop optimizer are.

$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) f^{'}(\theta_t)^2 \tag{2.19}$$

$$v_{t+1} = \frac{\alpha}{\sqrt{s_t}} f^{'}(\theta_t) \tag{2.20}$$

$$\theta_{t+1} := \theta_t - v_{t+1} \tag{2.21}$$

Adam Adaptive Moment Estimation *Adam* [43] is an advanced stochastic optimization algorithm employed to update the trainable parameters of the model which combines the best of both momentum and RMSprop. It is reported that in practice this results in better performance compared to the existing optimizer algorithms for a large variety of problems. Also, Adam has the least convergence time among the Gradient Decent optimizers. With v_t as the exponential average of gradient and s_t as the exponential average of squares of the gradients, the equations for the Adam optimizer are.

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * f^{'}(\theta_t) \tag{2.22}$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * f^{'}(\theta_t)^2 \tag{2.23}$$

While computing the first and second moment estimates, the initial values are computed by considering an arbitrary value v_0. Which results in bias towards v_0 while computing the initial steps of the exponentially weighted moving averages. This can be overcomed by performing bias correction. The equations to perform the bias correction on both the first moment estimate v_t and the second moment estimate s_t are as follows.

$$v_t^{corrected} = \frac{v_t}{1 - \beta_1^t} \tag{2.24}$$

$$s_t^{corrected} = \frac{s_t}{1 - \beta_2^t} \tag{2.25}$$

$$\theta := \theta - \alpha \frac{v_t^{corrected}}{\sqrt{s_t^{corrected}} + \epsilon} \tag{2.26}$$

The detail overview of the performance of the optimization of gradient descent algorithms are found in [51]

2.5.3 Regularization techniques

Dropout Dropout is a regularization technique first introduced by, which prevents the neural network from overfitting to the training set. In this technique, for each epoch, the network randomly drops the connections between the current layer and the next layer with a probability 'P.' Dropout layers are not used during inference, and the weights are multiplied with the probability 'P.'

Gaussian Noise Gaussian Noise is defined as statistical noise, generally added to the input feature vector to reduce the variance during training. The Probability Density Function of a Gaussian Noise is Gaussian. Hence it's named.

Batch Normalization Batch normalization also serves as a regularization technique is used to reduce the internal co-variance shift in the network and accelerate the training of neural nets. The equationa for batch-normalziation can be found in the figure 2.16.

2.5.4 Initializtion and Transformation Hyper-Parameters

Weight Initialization In training deep neural networks, for a long time, we had dealt with the problem of vanishing gradients and exploding gradients. Recent advances in the deep learning have resulted in different kinds of weights initialization techniques which resulted in a significant increase in the performance of a neural network.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Figure 2.16: Equations for Batch-Normalization[42]

In the scope of this thesis work the glorot uniform or Xavier uniform initializer[52] is used to initialize all the weights in the network. With $n^{[L-1]}$ representing the number of neuron in $L-1$ layer and $n^{[L]}$ denotes the number of neurons in L^{th} layer, the variance of initialization of the function parameters are as follows.

$$var(\theta_i) = \frac{2}{n^{[L-1]} + n^{[L]}} \tag{2.27}$$

Activation Functions As discussed earlier the core idea of Neural Networks algorithms is to approximate function in the given hypothesis space. However, irrespective of the depth of the architecture, neural networks with no activation, performs linear transformations to the input which results in a linear relationship between the input and output. The linear hypothesis space is highly limited and results in poor performance. Activation function transforms the input from linear hypothesis space into non-linear hypothesis space, resulting in significant improvement in performance. The activation functions used in the scope

of this thesis work include Rectified Linear Unit(ReLU), sigmoid, softmax and Exponential Linear Unit('ELU').

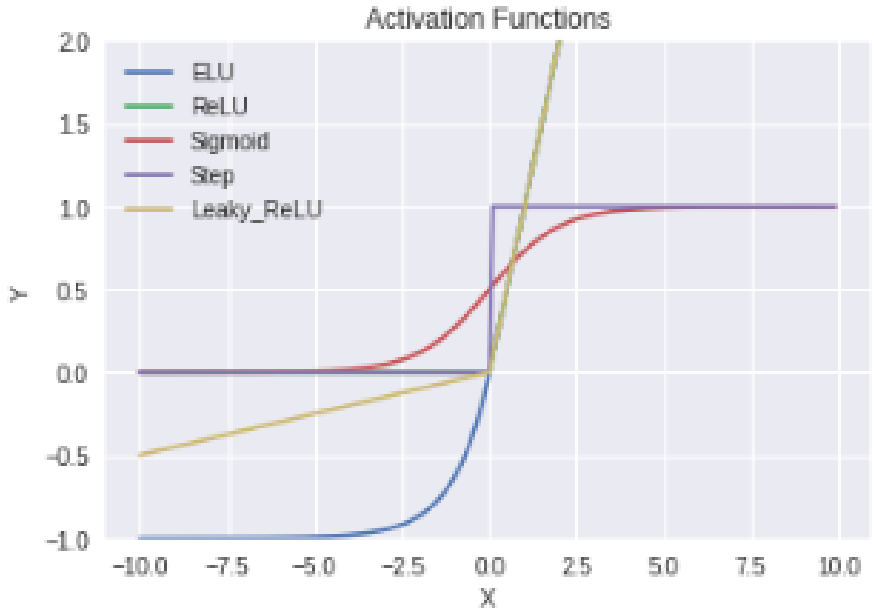


Figure 2.17: Activation Functions.

2.6 Principal Component Analysis

Principal Component Analysis[53] is a linear dimensionality reduction and visualization technique, used to map high-dimensional data with co-related features onto lower-dimensional space with uncorrelated features. This is achieved by calculating the new axes called principal components, such that the first few principal components retain the maximum variance existing in the original higher-dimensional space. In layman's terms, principal components are the directions in the Dataset with the highest variance.

The steps involved in performing the principal component analysis are as follows.

1) Pre-processing the dataset.

2) Calculating the co-variance matrix.

3) Performing Eigen Value Decomposition or Singular Value Decomposition on the co-

variance matrix.

4) Feature selection or Selecting the transformed features.

Preprocessing the dataset The Dataset needs to be normalized before performing PCA. The core idea of PCA is to map the original Dataset on to directions which maximize the variance. In the case of the unnormalized Dataset, if there exist some features with large variance and some with small variance, PCA is biased towards the features with the large variance, and the calculation of the principal components is dominated by such feature. Normalizing the Dataset by making each feature to stay on the same range will overcome this phenomenon.

Calculating the co-variance matrix Co-variance measures the strength of the relationship between two random variables. A positive covariance value represents that both the variables increase or decrease together, while negative value represents an inverse relationship between the two variables such that the increase or decrease in one variable results in decrease or increase in the value of the second feature, followed by zero co-variance represents two mutually independent variables.

$$cov(x, y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \tag{2.28}$$

Since covariance measures the strength of the relationship between two random variables when there exists more than two features or dimensions in the Dataset, the covariance matrix is calculated. For example consider a dataset with features x, y, and z, now the co-variance values are computed between x, y and, y, z and x, z. The co-variance calculation in commutative that is the co-variance between x, y, and y, x are equal. The covariance matrix is symmetrical across the diagonal with the diagonal representing the variances, which is the covariance of the variable with itself. A sample covariance matrix with the features x,

y and z is shown down below

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix} \tag{2.29}$$

In some instances, the correlation matrix is employed as an alternative to the covariance matrix. The correlation matrix is obtained by normalizing the covariance matrix by dividing it with the product of a standard deviation of the variables individually.

Performing Eigen Value Decomposition or Singular Value Decomposition Either of the Eigenvalue decomposition or Singular value decomposition is performed on the Covariance matrix to calculate the principal components. The eigenvectors corresponding to higher eigenvalue represents the stronger co-relation, while the eigenvectors corresponding to the lower eigenvalues represents weaker co-relation.

Feature selection or selecting the transformed features After calculating the principal components of the dataset, we have a choice of choosing then features contributing most to the calculation of the first few principal components in various combinations or can select the transformed features(principal components) as the new set of features for further processing

2.7 Random Forest Algorithm

Random Forests[54] algorithm, first proposed by Breiman in 2001, belongs to a class of ensemble methods employed for both classification and regression tasks. The random forest consists of a random combination of learning models called decision trees. This bagging architecture results in improved performance and reduced variance.

Similar to PCA, a random forest is also used to analyze the relative importance of the features in the dataset. When dealing with high dimensional datasets, in general, it is observed that there exists a fewer number of features which contribute to the accurate final prediction, while the other features add noise into the training algorithm and result in overfitting problem. The top 'N' features with higher relative importance can be selected and employed with better algorithms for further improvement in prediction accuracy.

As stated earlier, random forests are a collection of decision trees. However, there exist some critical differences between them. In random forest algorithm, each decision tree is built considering partial dataset and the arbitrary number of features; the final prediction is performed by taking a vote on the results obtained by all the decision trees in the random forest. However, decision trees are built on entire dataset resulting in significant amount of over-fitting problem.

2.8 Libraries

2.8.1 Kapre

Kapre [55] is a keras extension layer that could be placed as a part of the keras model. The feature extraction techniques like Spectrogram, Melspectrogram exists as a layer class which can be placed as a part of the keras model. Kapre layers allows to extract features on the fly on top of gpu.

2.8.2 librosa

Feature extraction for convolutional neural networks and visualization is done using the python package librosa [56].

2.8.3 openSMILE

Deep learning algorithms are hungry for data, The lack of availability of the data in the field of environmental sound classification, to some extent can be supplemented with increasing the dimensionality of the input data space, and this is achieved by using large feature extraction libraries like open Source Media Interpretation by Large feature Extraction (openSMILE). openSMILE was written in 'C' language and has support for live audio supported with Portaudio and openCV. Writing the custom configuration files in this environment lets us to extract a large hand engineered feature vector of a signal, and the data is stored in a CSV file. In one of the top performing model for DCASE task1a, a 6552dimensional feature vector was used to enhance the performance on a Deep neural network.

The feature extraction on an audio signal can be performed in three levels, in the first level, the features can be extracted from any point in the signal and are called instantaneous descriptors, followed by segmenting the signal into smaller frames of given size and extracting the features in the segmented regions or frames, and extracting features describing the relation between the features computed on multiple frames.

The following explains the various levels of audio feature extraction provided by the openSMILE library.

Low-Level audio descriptors(LLD's) Low-Level audio descriptors are computed by inspection of the audio signal, which represents the signal itself in various domains. The various domains in which the LLD's are calculated include

1) Temporal Descriptors

2) Spectral descriptors

3) Cepstral descriptors and

4) Perceptual descriptors

Delta Regression Coefficients In calculating the Low-Level Descriptors, the features are extracted from various points in the signal, and these features do not interact with each other. Delta-Regression coefficients are computed on LLD's as a post-processing step to calculate the relation between the features over frames. This gives a better understanding of the signal and also increases the model performance.

$$\delta_l^w(n) = \frac{\sum_{i=1}^{N} i * (x(n+i) - x(n-i))}{2 * \sum_{i=1}^{w} i^2} \tag{2.30}$$

Functionals: To perform the tasks of Automatic Speech Recognition, Music Genre Classification, human emotion classification, and Acoustic Scene Classification, the instantaneous features of the signal and the relation between them over frames are not sufficient. These tasks need to look into long-term dependencies, for example, to classify an acoustic scene the model needs to initially compute all the events happening in the scene, followed by identifying the relationship between those events to make the final prediction. To do this, statistical, polynomial, regression and transformations functionals are applied to the low-level features.

SMILE988 Features Extraction:

Emo_base[58] configuration file was employed to extract 988 features for each sample. The extracted features include 26 low-level descriptors (LLD) which contain 12 MFCC's, 8 line spectral frequencies, F_0 envelope, Intensity, Loudness, Pitch, probability of voicing, Zero-crossing rate. The low-level descriptors are used to compute 26 Delta regression coefficients followed by applying 19 functionals which include Max./Min. value of respective relative position within input, range, arithmetic mean, 2 linear regression coefficients and linear and quadratic error, standard deviation, skewness, kurtosis, quartile 1-3, and 3 inter-quartile ranges to low-level descriptors and Delta regression coefficients.

Feature Group	Description
Waveform	Zero-Crossings, Extremes, DC
Signal energy	Root Mean-Square & logarithmic
Loudness	Intensity & approx.loudness
FFT spectrum	Phase, magnitude(lin, dB, dBA)
ACF, Cepstrum	Autocorrelation and Cepstrum
Mel/Bark spectr.	Bands 0-N_mel
Semitone spectr.	FFT based and filter based
Cepstral	Cepstral features, e.g.MFCC,PLPCC
Pitch	F_0 via ACF and SHS methods
Voice Quality	HNR, Jitter, Shimmer
LPC	LPC coeff., reflect. coeff., residual Line spectral pairs(LSP)
Auditory	Auditory spectra and PLP coeff.
Formants	Centre frequencies and bandwidths
Spectral	Energy in N user-defined bands, multiple roll-off points, centroid,entropy, flux, and rel. pos. of max./min.
Tonal	CHROMA, CENS, CHROMA-based features

Table 2.3: [57] openSMILE's low-level descriptors

Category	Description
Extremes	Extreme values, positions, and ranges
Means	Arithmetic, quadratic, geometric
Moments	Std. dev., variance, kurtosis, skewness
Percentiles	Percentiles and percentile ranges
Regression	Linear and quad. approximation coefficients, regression err., and centroid
Peaks	Number of peaks, mean peak distance, mean peak amplitude
Segments	Number of segments based on delta thresholding, andmean segment length
Sample values	Values of the contour at configurable relative positions
Times/durations	Up- and down-level times, rise/fall times, duration
Onsets	Number of onsets, relative position of first/last on-/offset
DCT	Coefficients of the Discrete Cosine Transformation(DCT)
Zero-crossings	Zero-crossing rate, Mean-crossing rate

Table 2.4: [57] Functionals(Statistical, polynomial regression, and transformations) available in openSMILE

SMILE6k Feature Extraction:

Emo_large[59] configuration file was employed to extract 6552 features for each audio sample. This is the full scale extraction of features supported by openSMILE. The features include 56 low-level descriptors (LLD), 56 Delta regression features computed on LLD's and 39 functionals are applied on LLD'S and Delta Regression features.

The detailed description on the features extracted from the openSMILE library can be found in the chapter 2 of the document [60].

3 Performance Measures

There exists numerous performance metrics in the current machine learning literature. Defining the right metric is one of the crucial steps in defining the solution to the problem. The choice of the performance metric depends on various things which include, problem definition, the distribution of the classes in the dataset, and the trade-offs that we are consciously willing to make to yield an effective solution to the defined problem. The metrics employed in the scope of this thesis work include classification accuracy, precision, recall, Receiver Operating Characteristic curve, precision-Recall curves, and F1-score for multi-class with micro, macro, and weighted averages.

The naive approach to evaluating the performance of an binary or multi-class classifier is to calculate the classification accuracy. With P as number of correct predictions and N as the total number of samples in the dataset, classification accuracy is given by the ratio of P to N.

$$classification\,accuracy\ = \frac{Number\,of\,correct\,predictions(P)}{Number\,of\ samples\,in\,the\,dataset(N)} \tag{3.1}$$

$$Class\,wise\,classification\,accuracy\ = \frac{Number\,of\,predictions\,of\,class(p)}{Number\,of\ samples\,of\ class(p)} \tag{3.2}$$

		Actual Class		
		Back-Ground	Single Drone	Two Drone
Predicted Class	Back-Ground	100	0	0
	Single Drone	0	565	35
	Two Drone	0	51	49

Figure 3.1: Sample Confusion Matrix for multi-class classification

3.1 Confusion Matrix

The first step involved in the calculation of advanced performance metrics is to compute the confusion matrix. The confusion matrix consists of actual values on one dimension and predicted labels on the second dimension with each class consisting of a row and column. The diagonal elements of the matrix represents the correctly classified samples. The sample classification matrix for the multiple drone detection problem is shown in the figure 3.1.

The metrics that can be computed from the confusion matrix include Precision, Recall, F1-score which is the harmonic mean of the precision and recall, and the classification accuracy.

To build intuition on the metrics precision and recall, consider a binary classification problem with class-0 and class-1 corresponding to background noise and single drone in the scene.

Precision: It is defined as how many of our predicted samples with a drone in the scene actually contains the drone.

Recall: It is defined as the fraction of number of samples predicted to have a drone in the scene to the number of samples actually has drone in the scene.

F-Measure It is defined as the harmonic mean between the precision and recall.

3.2 Micro, Macro and Weighted Averages

Micro-average[61] is defined as the average of the same measures calculated for each of the classes. Macro-average[61] is defined as the sum of counts to obtain cumulative tp, fn, tn, fp and then calculating a performance measure.

When dealing with the classes with skewed class distribution, the micro-average technique is employed. In this, initially the F1-scores for each class is calculated, followed by the averaging the scores scaled with weights. The weights are generated the weighing it with the number of samples the corresponding class contains.

The following equations show the formula to compute the precision, recall and f1-

score in both the micro and macro-averaging techniques.

$$TP_i = True\ Positives\ of\ i^{th}\ class \quad (3.3)$$

$$TN_i = True\ Negatives\ of\ i^{th}\ class \quad (3.4)$$

$$FP_i = False\ Positives\ of\ i^{th}\ class \quad (3.5)$$

$$FN_i = False\ Negatives\ of\ i^{th}\ class \quad (3.6)$$

$$n = Total\ number\ of\ classes\ in\ the\ dataset \quad (3.7)$$

$$Precision_{macro-averaging} = \frac{\sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}}{n} \quad (3.8)$$

$$Recall_{macro-averaging} = \frac{\sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}}{n} \quad (3.9)$$

$$F - score_{macro-averaging} = 2 * \frac{Precision_{macro-averaging} * Recall_{macro-averaging}}{Precision_{macro-averaging} + Recall_{macro-averaging}} \quad (3.10)$$

$$Precision_{micro-averaging} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)} \quad (3.11)$$

$$Recall_{micro-averaging} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FN_i)} \quad (3.12)$$

$$F - score_{micro-averaging} = 2 * \frac{Precision_{micro-averaging} * Recall_{micro-averaging}}{Precision_{micro-averaging} + Recall_{micro-averaging}} \quad (3.13)$$

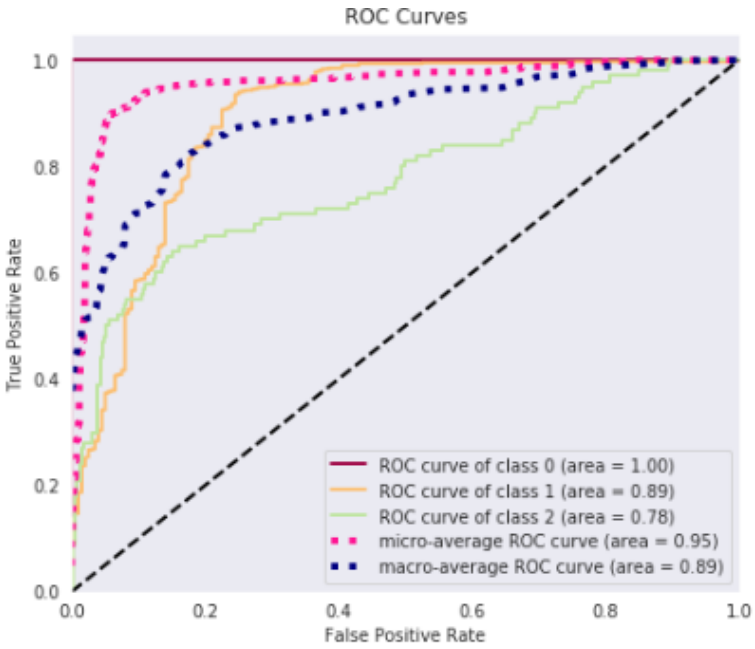
Unlike the cases of hard classification where the classifier outputs the class of a data sample, when soft classifiers are used which outputs the vector of probabilities representing it's confidence in belonging to a certain class, they are used to plot trade-off's between various metrics with varying thresholds. The Drone Detection dataset suffers from extremely skewed class distribution, so the classification accuracy provides no information gain in choosing the class. In order to individually evaluate the performance of the classifier for

each class, and also to compare the performance of various classifiers, in the following sections we discuss about the ROC and PR curves.

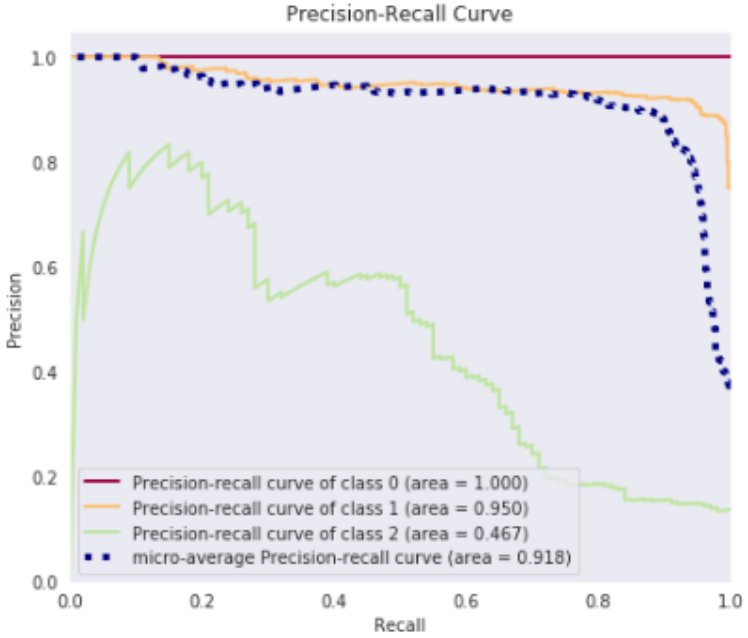
3.3 Receiver Operating Characteristic curve(ROC)

ROC curves are two dimensional plots representing the trade-off between the true positive rate(tp rate) and the false positive rate(fp rate) of a binary classifier for various thresholds ranging from 0 to 1. The fp-rate is represented by the x-axis and the tp-rate as a function of fp-rate is represented by the y-axis. For a intuitive understanding, consider a case of binary classification problem of detecting a drone in the scene. The lower the value of the threshold the higher the rate of drone detection will be, resulting in increase of false alarms. Higher threshold rates decreases the problem of having false alarms, and also decreases the rate of detection. By plotting the tp-rate and fp-rate for varying the thresholds over a range results in the ROC curve. The ROC curve can be analyzed in three components, the lower right triangle, diagonal, and upper left triangle. The point on the diagonal is interpreted as the classifier is making prediction at random gaining zero knowledge from training. The point in the lower right triangle implies that the classifier has gained some knowledge, however, the performance is poor. It is desired for the point to be located in the upper left triangle and closer the final curve closer to the upper border of the ROC curve. The area under the curve represents the performance of the classifier. The sample plot for the ROC curve is shown in the figure 3.2(a). ROC is insensitive to the skewed class distributions up to certain extent and can be plotted to multi-class classification with the one vs rest classifier approach. However, [62] states that ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the dataset, and PR-curves are an alternative to ROC-curve in the presence of large skew in the class distribution. In the problem of Multiple Drone Detection it is necessary to understand the trade-off between the precision and recall, also, since the drone detection dataset has extremely skewed class

distribution, In the scope of this thesis work Precision-Recall curves are employed for evaluating the performance of a classifier.



(a) ROC-Curve



(b) PR-Curve

Figure 3.2: PR and ROC plots

3.4 Precision-Recall Curves

Precision-Recall curves are a two-dimensional representation of the trade-off between the precision and recall values for various thresholds of a classifier, with x-axis representing recall and the y-axis representing the precision. The PR-curves are highly insensitive to skewed class distributions and are easily modeled for the multi-class scenarios. Unlike the monotonically increasing ROC-curve, PR-curve can move in either direction. The figure 3.2 shows the resultant plots for ROC and PR curves for a sample experiment performed on the drone detection dataset. The class wise classification accuracy for class 2 is 49 percent. However, ROC modeled the AUC for 0.78 which is misleading, where as from the PR-curve the AUC for the class 2 is 0.48 which is quite accurate. Also, the comparison between the classifiers can be effectively done by looking at the AUC for the PR-curves.

In the experiments performed in the following chapters, for each epoch the weighted average f1-score for the validation set is monitored and the state of the model yielding better performance on this metric is saved for further calculations and plotting the necessary functions. The PR-curves, classification report with weighted average and normalized confusion matrix, calculated from the average performance of the best performing model is plotted for every experiment in the following chapters.

4 Experiments: Drone Detection

4.1 Multiple Drone Detection

This section deals with the experiments performed on the problem of multiple drone detection with the custom collected dataset. The experiments in this chapter are organized in the following order.

1) PCA and TSNE visualization of the SMILE988 features.

2) Random Forest Algorithm applied to the SMILE988 features.

3) Deep Neural Network with SMILE988 features.

4) Deep Neural Network with SMILE988 reduced features

5) Convolutional Neural Network with 3-channel spectrograms.

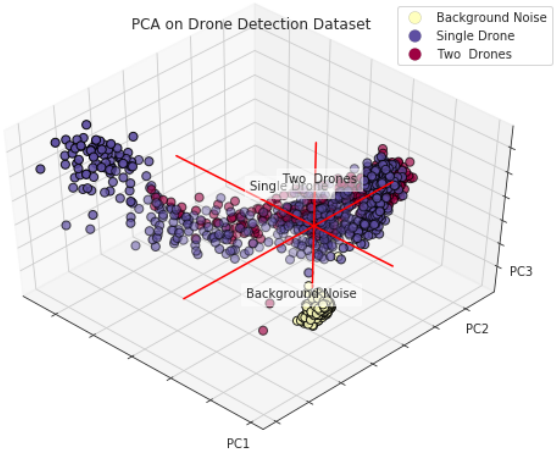
6) Convolutional Neural Network with 2- channel Spectrograms with Harmonic and Percussive content into individual channels.

7) Convolutional Neural Network with Raw audio waveforms.

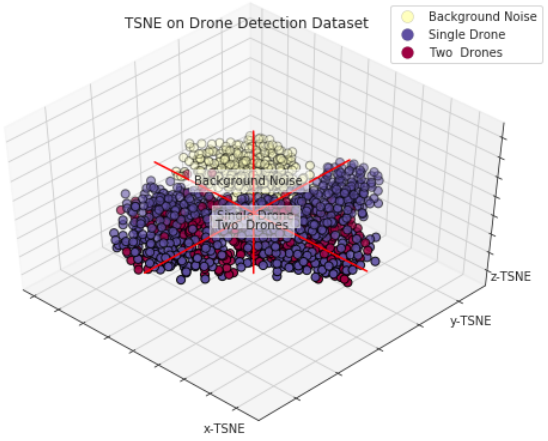
8) Generative Adversarial Networks for Data Augmentation.

4.2 Experiment 1: PCA and TSNE visualization

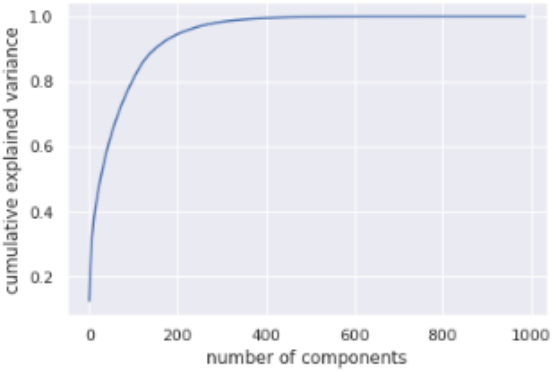
In this experiment, using the openSMILE large-scale feature extraction library, 988 features were extracted for each audio sample in the dataset. The extracted features are later formatted into a single CSV file. The brief description of the feature extraction scheme and the link to the complete documentation of the library can be found in the section 2.8.3 of this document. In the next step, the features are normalized with a zero mean and unit standard deviation. Followed by the application of linear and non-linear dimensionality reduction techniques like principal component analysis and the t-Distributed Stochastic Neighbor Embeddings on the smile988 features extracted on the drone detection dataset. The objective is to identify the most important features that contribute to the increase in the performance of the classifier, among the extracted features. The PCA analysis can be used to perform the feature selection and feature transformation. In this experiment, feature selection is made by identifying the set of 200 important features that contribute the most in the calculation of the first three principal components. The visualization of PCA and TSNE in three-dimensional space is shown in figure 4.1. Also, the top 15 features contributing to the calculation of the first three principal components are shown in table 4.1.



(a) PCA Visualization



(b) TSNE visualization



(c) Cummulative explained Variance

Figure 4.1: SMILE988 Data Visualization for Drone Detection dataset

Principal Component 1	Principal Component 2	Principal Component 3
F0_sma_linregerrQ	F0env_sma_linregerrQ	F0_sma_de_linregerrQ
F0env_sma_linregerrQ	F0_sma_linregerrQ	F0_sma_linregerrQ
F0_sma_de_linregerrQ	F0_sma_de_linregerrQ	F0env_sma_quartile1
F0_sma_range	F0_sma_max	F0_sma_max
F0_sma_max	F0_sma_range	F0_sma_range
F0env_sma_max	F0env_sma_range	F0env_sma_linregc2
F0env_sma_range	F0env_sma_linregc2	F0env_sma_quartile2
F0env_sma_quartile3	F0env_sma_max	F0_sma_quartile2
F0env_sma_quartile2	F0env_sma_iqr1-3	F0env_sma_amean
F0env_sma_amean	F0_sma_quartile3	F0env_sma_quartile3
F0env_sma_quartile1	F0env_sma_stddev	F0env_sma_range
F0_sma_quartile3	F0_sma_iqr1-3	F0env_sma_max
F0_sma_de_range	F0_sma_de_range	F0_sma_de_range
F0_sma_iqr1-3	F0env_sma_iqr2-3	F0env_sma_linregerrQ
F0env_sma_linregc2	F0_sma_linregc2	F0_sma_linregc2

Table 4.1: Most Contributing Features for First 3-Principal Components for Drone detection dataset

4.3 Experiment 2: Random Forest Algorithm with SMILE988 Features

In PCA analysis, we have the most important features that contribute to the calculation of the first three-Principal components. However, PCA deals with the features as a group with a weighted average. To understand the information gain provided by the each individual feature in the dataset, random forest regressor is employed on the SMILE988 feature set. The regressor is trained with 2500 estimators, resulting in the classification accuracy of 73.3 percent. The top features set is populated by the line spectral pairs, enerygy features, and MFCC's.

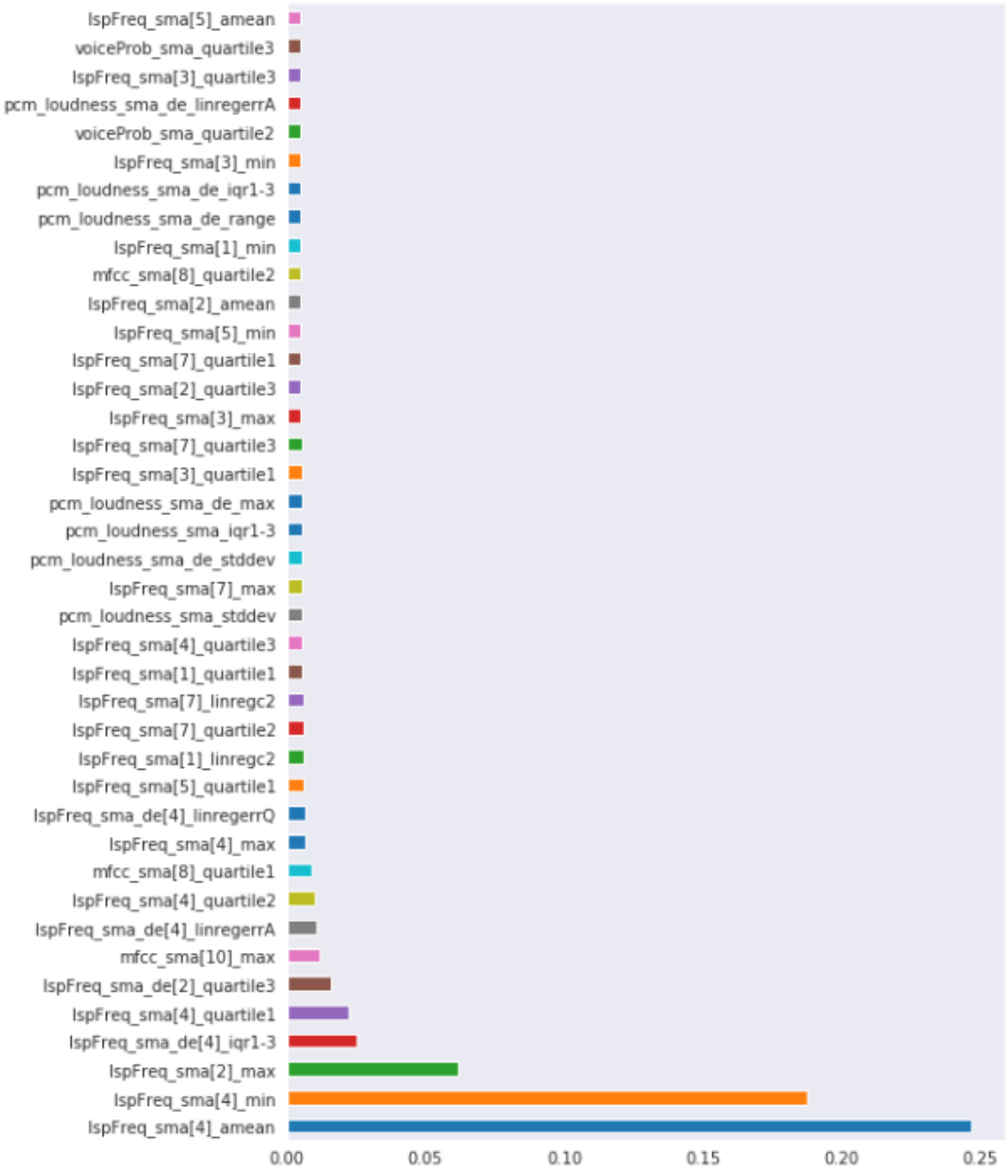


Figure 4.2: 40 Most Contributing features for the Random Forest algorithm

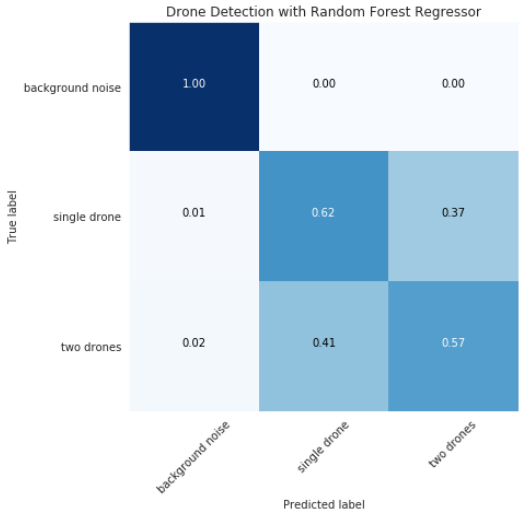


Figure 4.3: Confusion Matrix for Random Forest algorithm

4.4 Experiment 3: DNN with SMILE988 Features

In this experiment, a two-layered deep neural network is trained with the smile988 features. During training Gaussian Noise with a standard deviation of 0.02 is added to the input feature vector for improving the generalization performance. The network is trained for 300 epochs with a batch size of 16 samples. The performance is evaluated with a categorical accuracy loss function and the micro-averaged Area Under Curve(AUC) of PR-Curve. The model resulting in the highest AUC on the validation set is saved aside into the final inference model. This experiment yielded a total accuracy of 84.6 percent. The additional hyper-parameters involved include Elu units for the activation layers, kernel regularization is applied at every layer. Variants of architectures have been experimented by increasing the depth of the network, increasing the number of neurons in the dense layers, and tuning the hyper-parameters. Of which the two-layered network architecture shown in figure 4.4 resulted in the best performance of the model on average. The hyper-parameters employed in experimenting are shown in table 4.2. This experiment resulted in a classification accuracy of 84.2 percent, AUC for micro-average PR-curve is 0.88, and micro-average F1-score

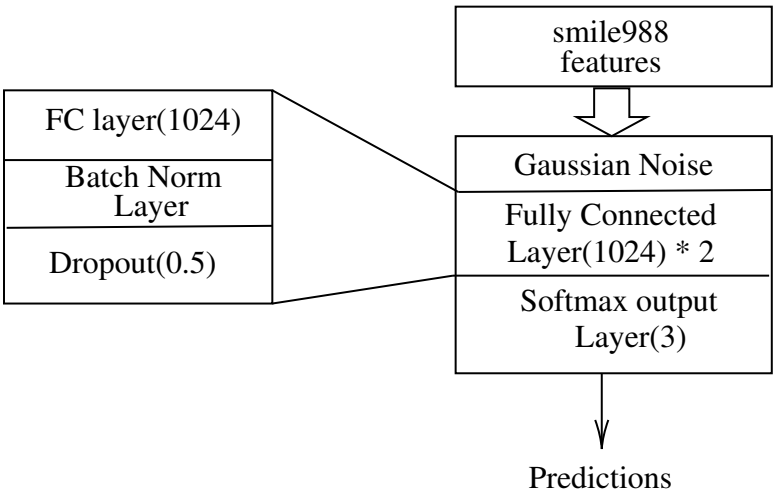
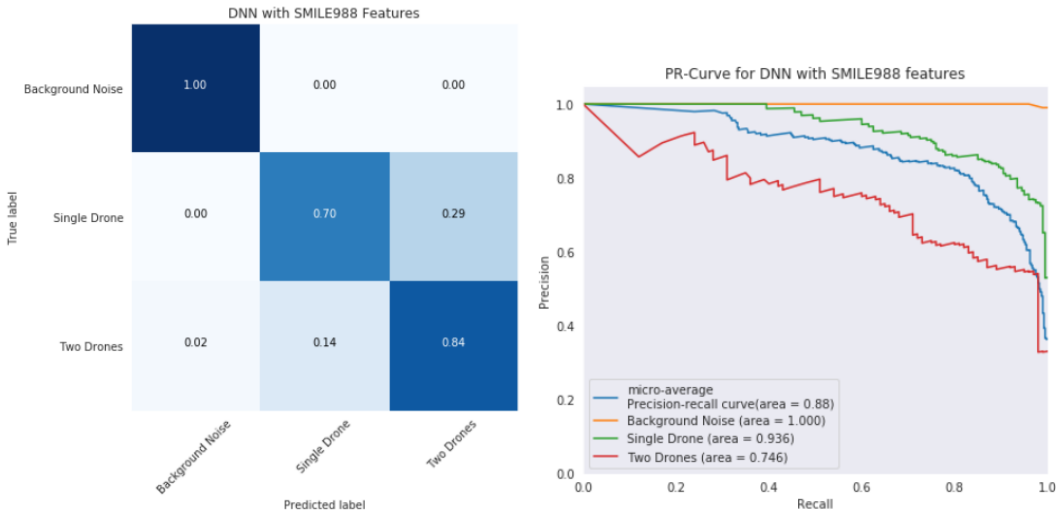


Figure 4.4: DNN with smile988

Feature Employed	Smile988
Activation Function	Exponential Linear Unit
Batch Size	16
Optimizer	Rmsprop
Metrics	AUC(PR-curve)$_{micro-average}$
Loss Function	Categorical Cross Entropy
Dataset	Oringinal Dataset

Table 4.2: Hyper Parameters

is 0.82.



(a) Confusion Matrix (b) PR-Curve

Figure 4.5: DNN with SMILE988 Results

Classes	Precision	Recall	F1-Score
Background Noise	0.98	1	0.99
Single Drone	0.91	0.70	0.79
Two Drones	0.59	0.84	0.69
avg/total	0.85	0.81	0.82

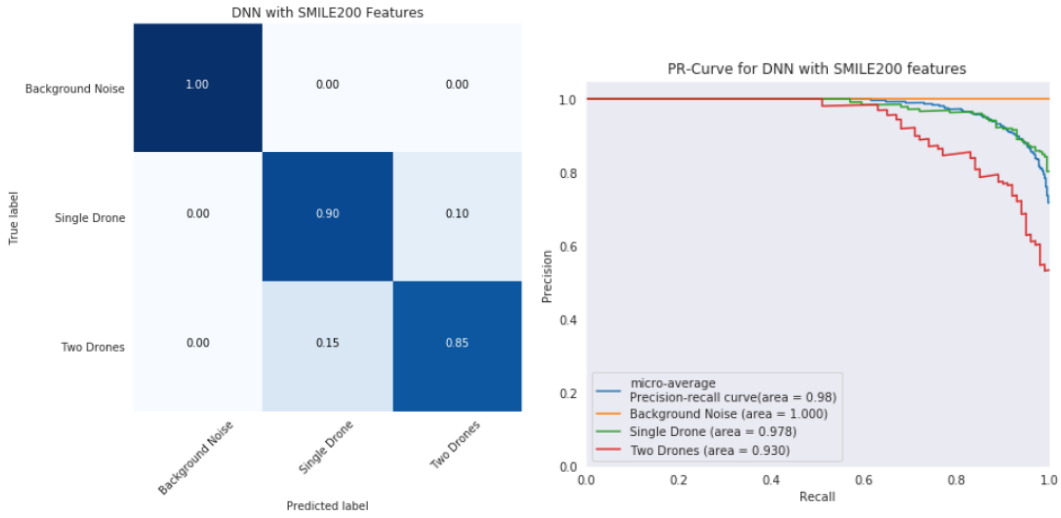
Table 4.3: Classification Report for DNN with SMILE988 Features

The results are descent for this experiment, and the PR-curve for each class are nearer to the upper right angle part, which is desired.

4.5 Experiment 4: DNN with SMILE200 features

In this experiment, the subset of SMILE988 features of about 200 features, contributing most to the calculation of the Principal components, are fed into the two layered-DNN for training. The network was trained for 400 epochs, with the similar environment employed in the experiment 3 of this chapter. The SMILE200 feature selection scheme has improved the average performance of the system by 7 percent over DNN with SMILE988. This

experiment resulted in classification accuracy of 91.3 percent, AUC for micro-average PR-curve is 0.98, and micro-average F1-score is 0.91.



(a) Confusion Matrix (b) PR-Curve

Figure 4.6: DNN with SMILE200 Results

Classes	Precision	Recall	F1-Score
Background Noise	1	1	1
Single Drone	0.92	0.90	0.91
Two Drones	0.81	0.85	0.83
avg/total	0.91	0.91	0.91

Table 4.4: Classification Report for DNN with SMILE200 Features

4.6 Experiment 5: CNN with Spectrograms

Here a series of experiments are performed employing the variants of spectrograms, in their raw forms and on the psycho-acoustic frequency scales. The extracted sets of features include raw spectrograms, Log-spectrograms, Mel-Spectrograms, and Log-Mel Spectrograms. For Log-Mel spectrograms 4 feature sets are extracted for different number of Mel filter bank's. Mels-spectrograms are been the standard choice of spectro-temporal representation of audio data in the tasks of Sound Event Detection and Automatic Speech

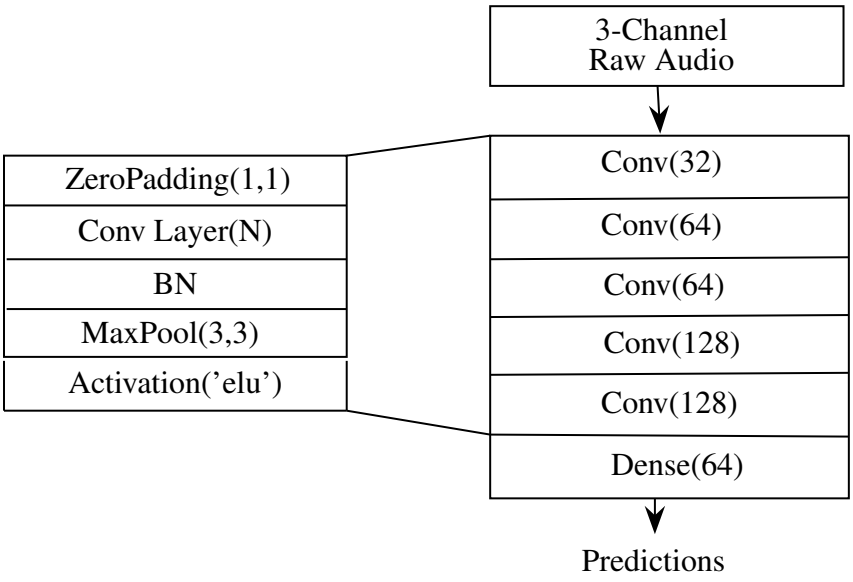


Figure 4.7: CNN with Spectrograms

Features	Num Channels	Input Shape	Window Size	Hop Length
Spectrograms	Left, Right, Left - Right	(1025, 87, 3)	2048	512
Log-Spectrograms	Left, Right, Left - Right	(1025, 87, 3)	2048	512
Log-Mel Spectrograms with 40 Mel filters	Left, Right, Left - Right	(40, 87, 3)	2048	512
Log-Mel Spectrograms with 60 Mel Filters	Left, Right, Left - Right	(60, 87, 3)	2048	512
Log-Mel Spectrograms with 128 Mel Filters	Left, Right, Left - Right	(128, 87, 3)	2048	512
Log-Mel Spectrograms with 200 Mel Filters	Left, Right, Left - Right	(200, 87, 3)	2048	512

Table 4.5: Input Feature Vector for CNN with Spectrograms

Recognition, however, considering the fact that the Mel spectrograms are designed to imitate the human auditory system, and the human performance is poor in identifying the multiple overlapping events. We have the prior assumption that, mel spectrograms may lose some important information when dealing with the problem of building an audio pattern recognition systems. The architecture for CNN employed in this experiment can be found in the figure 4.7. The hyper parameters employed for all the trials have been listed

Feature Employed	Smile988
Activation Function	Exponential Linear Unit
Batch Size	16
Optimizer	Rmsprop
Metrics	Categorical accuracy
Loss Function	Categorical Cross Entropy
Dataset	augmented Dataset

Table 4.6: Hyper Parameters for CNN with Spectrograms

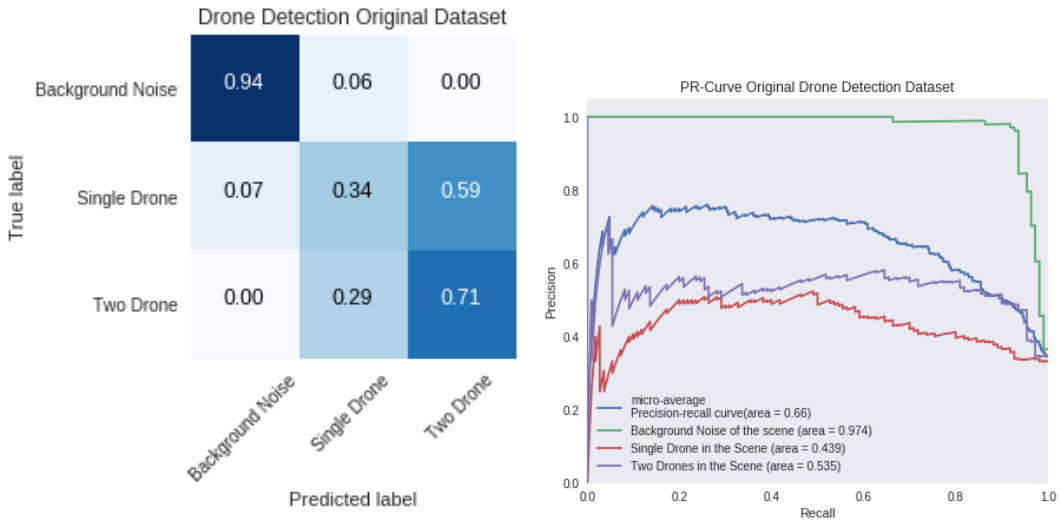
in the table 4.6.

4.6.1 CNN with Raw Spectrograms

The stereo audio consists of two channels left and right, and the third channel is obtained by subtracting the channel right from the channel left. The STFT is performed on the three channels individually with a frame size of 2048 samples and a hop length of 512 samples. This resulted in a three-channel raw spectrogram. The 5-layer CNN shown in figure 4.7 is trained for 200 epochs with a batch size of 16, and the RmsProp optimizer is used with the categorical cross-entropy loss function, and the exponential linear units are employed in the activation layer, except for the output layer. This experiment resulted in a classification accuracy of 66.3 percent, AUC for micro-average PR-curve is 0.66, and micro-average F1-score is 0.65. ‘

4.6.2 CNN with Log-Spectrograms

In linear domain the content of the spectrograms are concentrated mostly around the lower end of the frequency spectrum, in log-domain, the content is distributed with equal weights across the image. This experiment deals with the 3-channel raw spectrograms obtained from the previous experiment in logarithmic domain. In log-domain equal preference is shown on each individual frequency bin of the spectrogram. The prior assumptions are to observe a increase in performance compared to the ones in the linear scale. The training



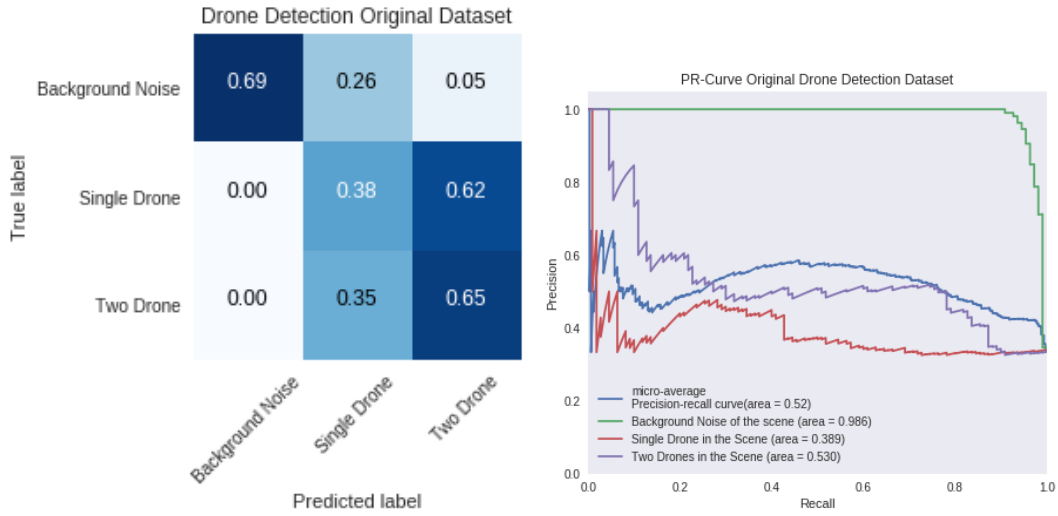
(a) Confusion Matrix (b) PR-Curve

Figure 4.8: CNN with raw Spectrograms Results

Classes	Precision	Recall	F1-Score
Background Noise	0.93	0.94	0.93
Single Drone	0.49	0.34	0.40
Two Drones	0.55	0.71	0.62
avg/total	0.65	0.66	0.65

Table 4.7: Classification Report for CNN with Spectrograms

is performed for a period of 200 epochs with a batch size of 16. This experiment resulted in classification accuracy of 57.33 percent, AUC for micro-average PR-curve is 0.52, and micro-average F1-score is 0.59.



(a) Confusion Matrix (b) PR-Curve

Figure 4.9: CNN with Log-Spectrograms Results

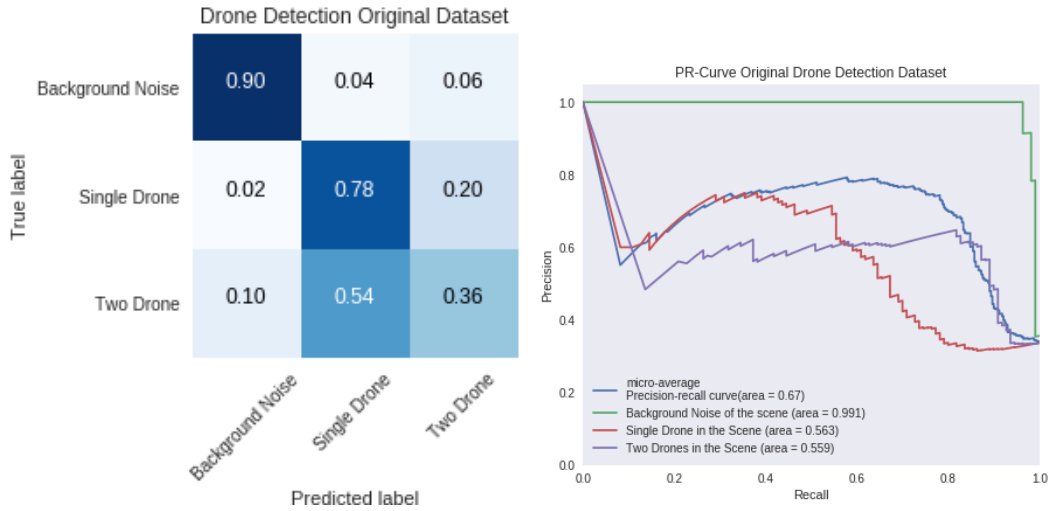
Classes	Precision	Recall	F1-Score
Background Noise	1	0.69	0.82
Single Drone	0.39	0.38	0.38
Two Drones	0.50	0.65	0.56
avg/total	0.63	0.58	0.59

Table 4.8: Classification Report for CNN with Log-Spectrograms

4.6.3 CNN with Mel-Spectrograms with 128 Mels

For the 3-channel Spectrograms extracted in the previous experiment. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 5-layer CNN is trained for 200 epochs with batch size of 16. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 68 percent, AUC for micro-average PR-

curve is 0.67, and micro-average F1-score is 0.67.



(a) Confusion Matrix (b) PR-Curve

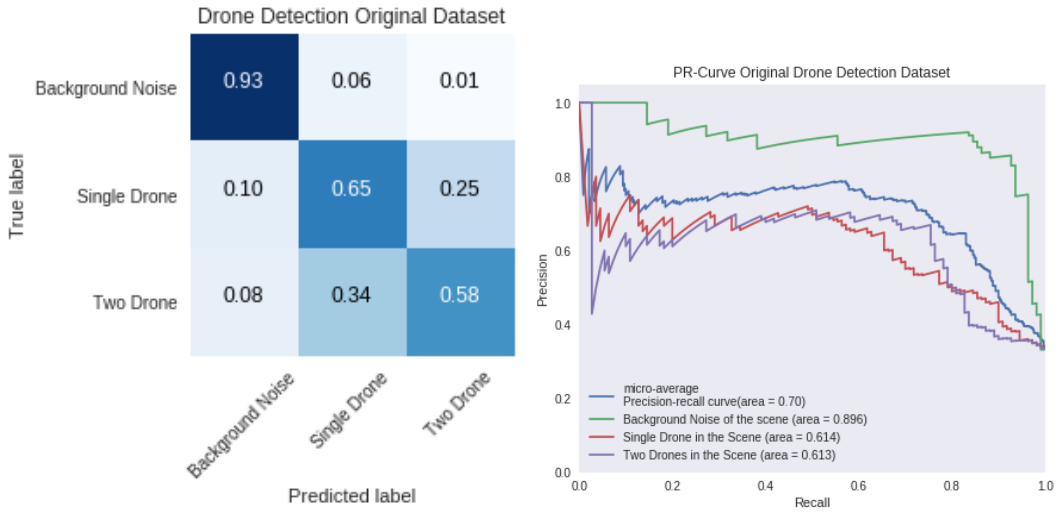
Figure 4.10: CNN with Mel-Spectrograms with 128 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.88	0.90	0.89
Single Drone	0.58	0.78	0.66
Two Drones	0.58	0.36	0.45
avg/total	0.68	0.68	0.67

Table 4.9: Classification Report for CNN with Mel-Spectrograms with 128 Mels

4.6.4 CNN with Log-Mel Spectrograms with 40 Mels

The last convolutional block in the CNN architecture shown in the figure 3.4 is popped out for this experiment, since it's presence resulting in negative dimensions. with the rest of the architecture intact, 3-channel Log-Mel Spectrograms extracted by 40 Mel filters are fed into the CNN. The network is trained for 200 epochs with a batch size of 128. This experiment resulted in classification accuracy of 72 percent, AUC for micro-average PR-curve is 0.70, and micro-average F1-score is 0.72.



(a) Confusion Matrix (b) PR-Curve

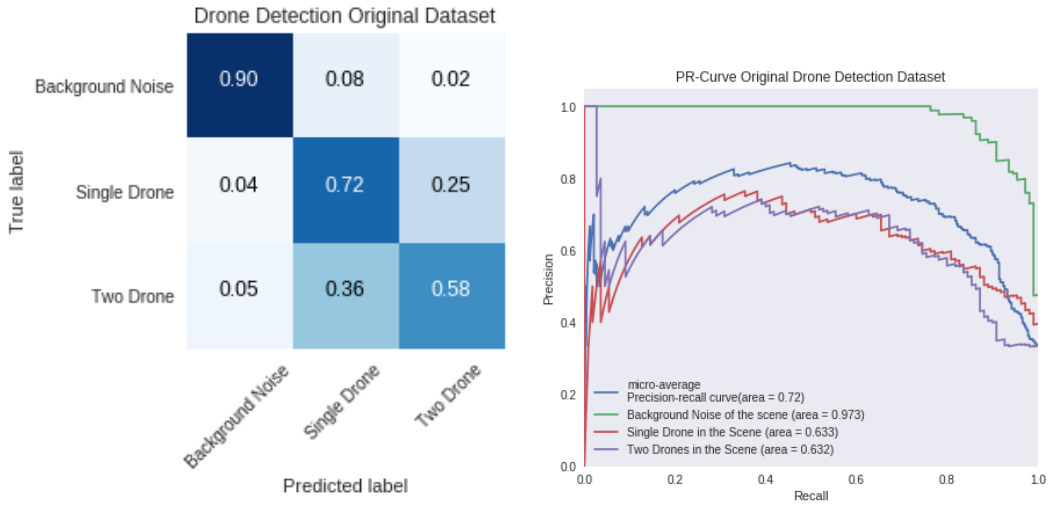
Figure 4.11: CNN with Log-Mel Spectrograms with 40 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.84	0.93	0.88
Single Drone	0.62	0.65	0.64
Two Drones	0.70	0.58	0.63
avg/total	0.72	0.72	0.72

Table 4.10: Classification Report for CNN with Log-Mel Spectrograms with 40 Mels

4.6.5 CNN with Log-Melspectrograms with 60 mel filters

The 3-channel Log-Melspectrograms are extracted by employing a Mel filter bank with 60 Mel filters over the raw spectrograms. The network is trained for 200 epochs with a batch sizer of 128. This experiment resulted in classification accuracy of 73.33 percent, AUC for micro-average PR-curve is 0.72, and micro-average F1-score is 0.73.



(a) Confusion Matrix (b) PR-Curve

Figure 4.12: CNN with Log-Mel Spectrograms with 60 Mels Results

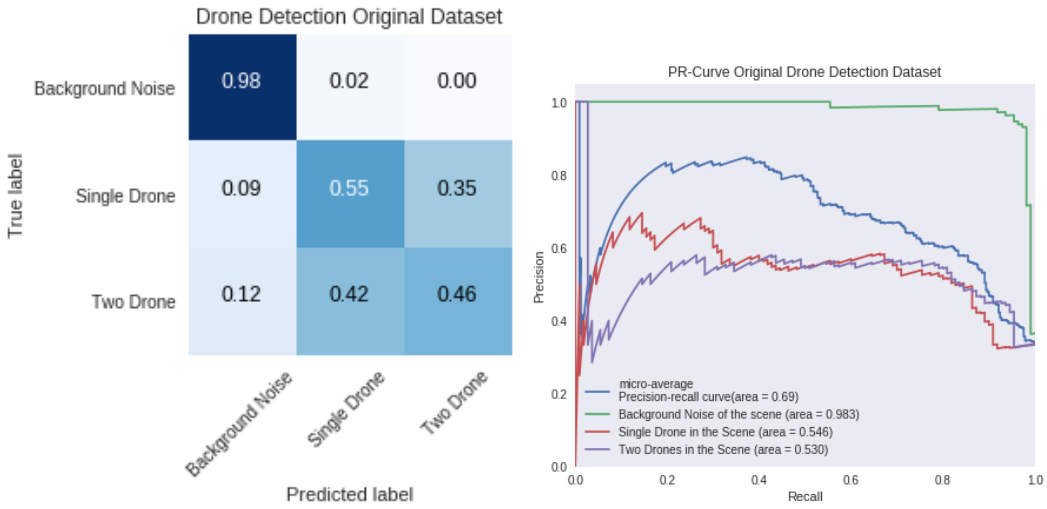
Classes	Precision	Recall	F1-Score
Background Noise	0.91	0.90	0.90
Single Drone	0.62	0.72	0.66
Two Drones	0.69	0.58	0.63
avg/total	0.74	0.73	0.73

Table 4.11: Classification Report for CNN with Log-Mel Spectrograms with 60 Mels

4.6.6 CNN with Log-Mel Spectrograms with 80 Mels

In this experiment 3-channel Log-Mel Spectrograms are obtained by filtering the three-channel raw spectrograms through a mel filter bank with 80 Mel filters. The network is

trained for a period of 200 epochs, with 128 samples per batch. This experiment resulted in classification accuracy of 66.3 percent, AUC for micro-average PR-curve is 0.69, and micro-average F1-score is 0.65.



(a) Confusion Matrix (b) PR-Curve

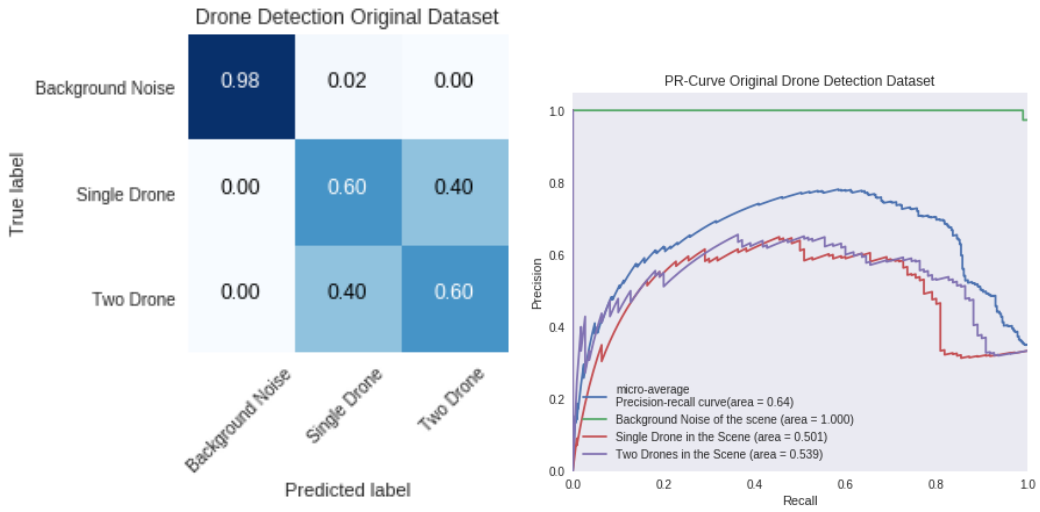
Figure 4.13: CNN with Log-Mel Spectrograms with 80 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.82	0.98	0.90
Single Drone	0.56	0.55	0.56
Two Drones	0.57	0.46	0.51
avg/total	0.65	0.67	0.65

Table 4.12: Classification Report for CNN with Log-Mel Spectrograms with 80 Mels

4.6.7 CNN with Log-Mel Spectrograms with 128 Mels

In this experiment 3-channel Log-Mel Spectrograms are obtained by filtering the three-channel raw spectrograms through a mel filter bank with 128 Mel filters. The network is trained for a period of 200 epochs, with 128 samples per batch. This experiment resulted in classification accuracy of 72.7 percent, AUC for micro-average PR-curve is 0.64, and micro-average F1-score is 0.73.



(a) Confusion Matrix (b) PR-Curve

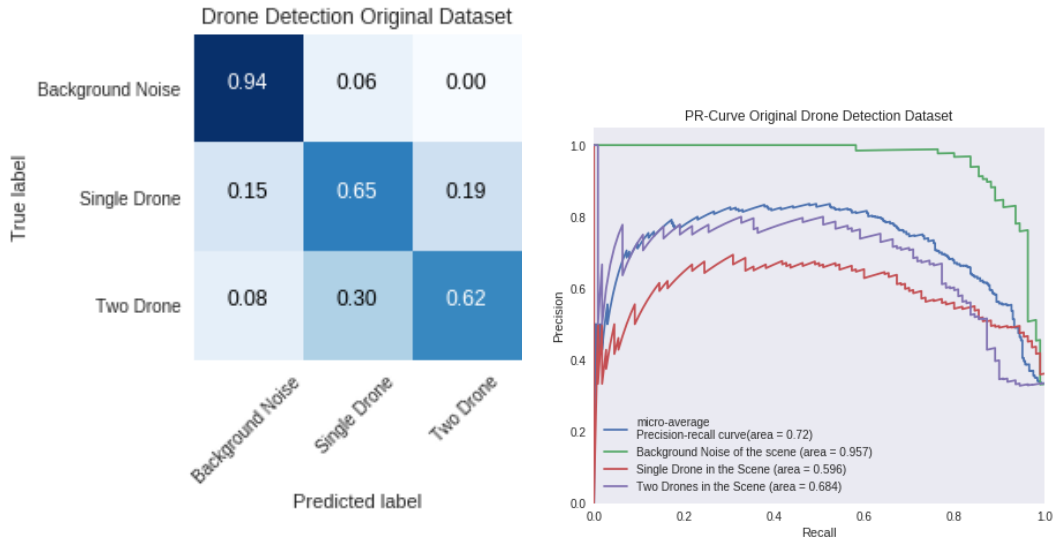
Figure 4.14: CNN with Log-Mel Spectrograms with 128 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	1	0.98	0.99
Single Drone	0.59	0.60	0.59
Two Drones	0.60	0.60	0.60
avg/total	0.73	0.73	0.73

Table 4.13: Classification Report for CNN with Log-Mel Spectrograms with 128 Mels

4.6.8 CNN with Log-Mel Spectrograms with 200 Mels

In this experiment 3-channel Log-Mel Spectrograms are obtained by filtering the three-channel raw spectrograms through a mel filter bank with 200 Mel filters. The network is trained for a period of 200 epochs, with 128 samples per batch. This experiment resulted in classification accuracy of 73.7 percent, AUC for micro-average PR-curve is 0.72, and micro-average F1-score is 0.73.



(a) Confusion Matrix (b) PR-Curve

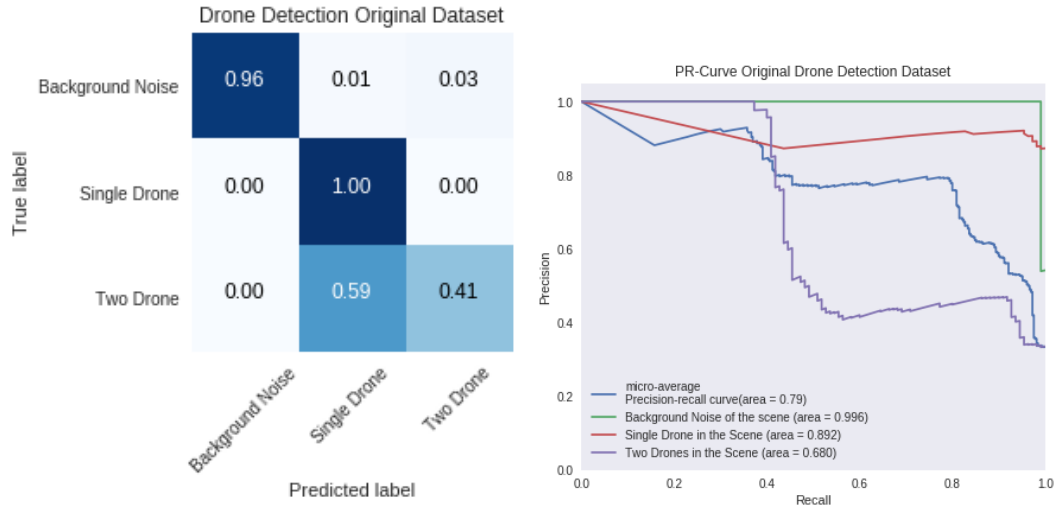
Figure 4.15: CNN with Log-Mel Spectrograms with 200 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.80	0.94	0.86
Single Drone	0.64	0.65	0.65
Two Drones	0.76	0.62	0.68
avg/total	0.74	0.74	0.73

Table 4.14: Classification Report for CNN with Log-Mel Spectrograms with 200 Mels

4.7 Experiment 6: CNN with Harmonic Percussive Source Separation

In the previous experiments with spectrograms, it is observed that the performance of the raw spectrograms and its variants employed with CNN did not improve the performance of the classifier. In this experiment the spectrograms are extracted over a monophonic audio, and harmonic and percussive components of the sound are filtered into two individual channels. The network is trained for 200 epochs with 16 samples per batch. This experiment resulted in classification accuracy of 79 percent, AUC for micro-average PR-curve is 0.79, and micro-average F1-score is 0.77.



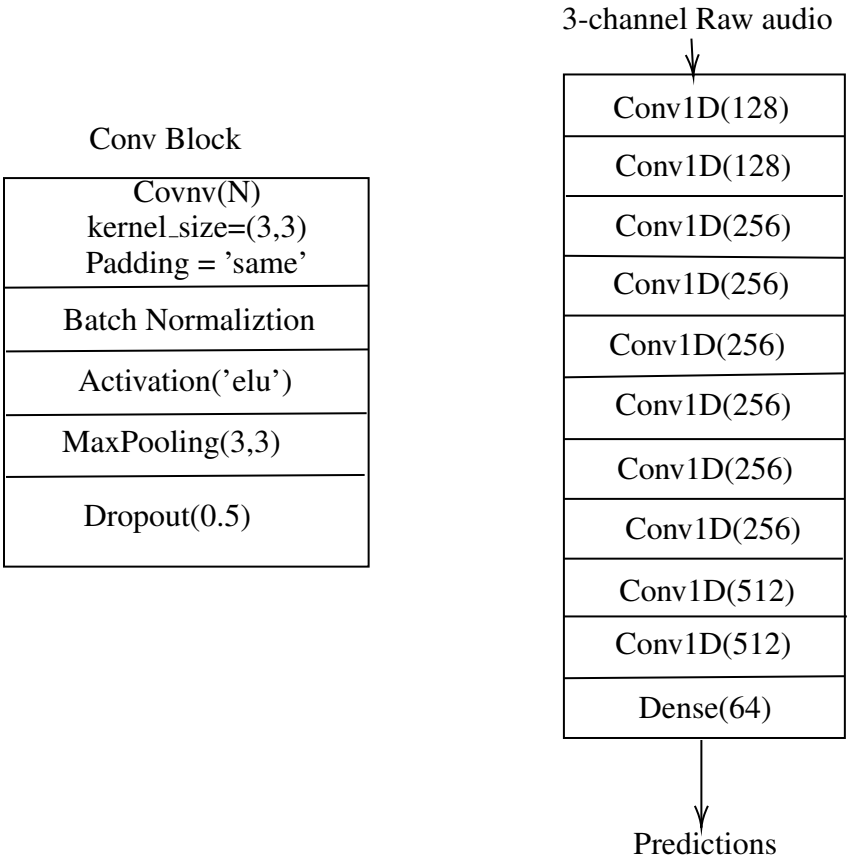
(a) Confusion Matrix (b) PR-Curve

Figure 4.16: CNN with Harmonic-Percussive Source Separation Results

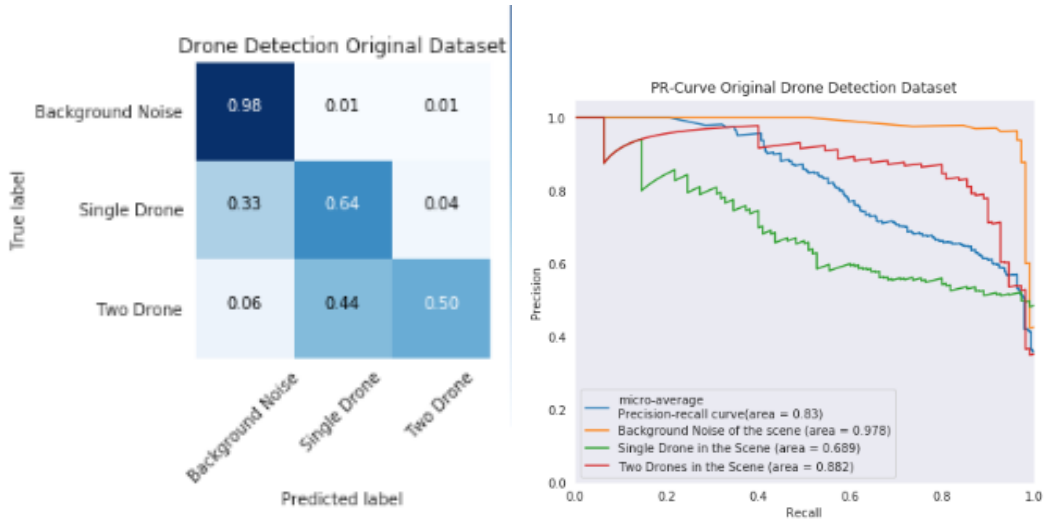
Classes	Precision	Recall	F1-Score
Background Noise	1	0.96	0.98
Single Drone	0.62	1	0.77
Two Drones	0.94	0.41	0.57
avg/total	0.85	0.79	0.77

Table 4.15: Classification Report for CNN with Harmonic-Percussive Source Separation

4.8 Experiment 7: CNN with Raw Audio Forms



Deep Neural Networks have achieved state of the art results performing on raw data. This eliminates the human bias to the solution. [63] have shown the sample-level CNN architecture employed with raw audio waveforms yielding state of art performance on environmental sound classification datasets. The left channel, right channel, left - right channels are used as a three channel input to the CNN. This experiment resulted in total accuracy of 66 percent. The network is trained with rmsprop optimization algorithm, and categorical loss function. The network is trained for 200 epochs with a batch size of 1. This experiment resulted in a accuracy of 70 percent. The observation are, unlike training with melspectrograms and Spectrograms the the results are consistent. This experiment resulted in classification accuracy of 70.6 percent, AUC for micro-average PR-curve is 0.83, and micro-average F1-score is 0.70.



(a) Confusion Matrix (b) PR-Curve

Figure 4.17: CNN with 3-channel Raw audio waveforms Results

Classes	Precision	Recall	F1-Score
Background Noise	0.72	0.99	0.83
Single Drone	0.59	0.64	0.61
Two Drones	0.92	0.50	0.65
avg/total	0.74	0.71	0.70

Table 4.16: Classification Report for CNN with 3-channel Raw-audio Waveforms

4.9 Drone Detection Dataset with Generative Models

The Generative Adversarial Networks[64] has shown astounding results in the field of computer vision. Recent advances[65] in ASR have resulted in speech audio generation which can be listened by the humans. Followed by [66] have shown that GAN can be used to generate the raw audio. In this experiment the GAN architecture employed in [66] is used for generation of the Drone Audio. The main objective of this experiment is to generate new audio for the classes single drone in the scene and two drones in the scene to increase the size of the dataset to train Deep architectures.

The network is trained with the audio samples from the class single drone in the scene, after 1000 epochs of training the model started generating human perceivable audio for the

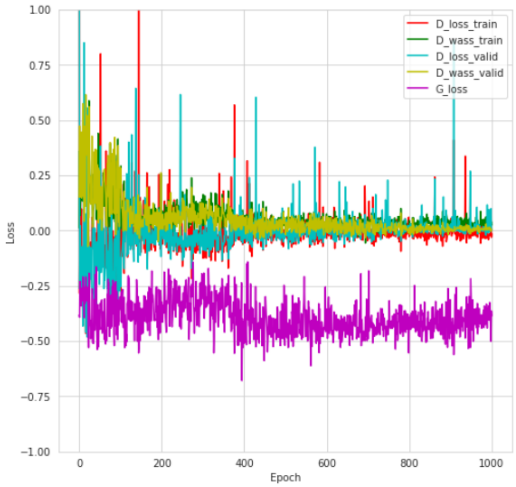
Table 3. WaveGAN generator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{z} \sim \text{Uniform}(-1, 1)$		$(n, 100)$
Dense 1	$(100, 256d)$	$(n, 256d)$
Reshape		$(n, 16, 16d)$
ReLU		$(n, 16, 16d)$
Trans Conv1D (Stride=4)	$(25, 16d, 8d)$	$(n, 64, 8d)$
ReLU		$(n, 64, 8d)$
Trans Conv1D (Stride=4)	$(25, 8d, 4d)$	$(n, 256, 4d)$
ReLU		$(n, 256, 4d)$
Trans Conv1D (Stride=4)	$(25, 4d, 2d)$	$(n, 1024, 2d)$
ReLU		$(n, 1024, 2d)$
Trans Conv1D (Stride=4)	$(25, 2d, d)$	$(n, 4096, d)$
ReLU		$(n, 4096, d)$
Trans Conv1D (Stride=4)	$(25, d, c)$	$(n, 16384, c)$
Tanh		$(n, 16384, c)$

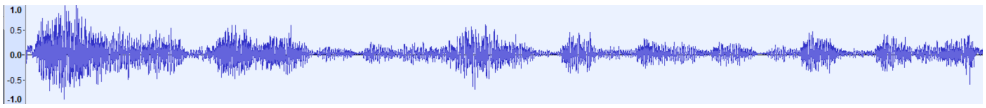
Table 4. WaveGAN discriminator architecture

Operation	Kernel Size	Output Shape
Input $\boldsymbol{x}$ or $G(\boldsymbol{z})$		$(n, 16384, c)$
Conv1D (Stride=4)	$(25, c, d)$	$(n, 4096, d)$
LReLU ($\alpha = 0.2$)		$(n, 4096, d)$
Phase Shuffle ($n = 2$)		$(n, 4096, d)$
Conv1D (Stride=4)	$(25, d, 2d)$	$(n, 1024, 2d)$
LReLU ($\alpha = 0.2$)		$(n, 1024, 2d)$
Phase Shuffle ($n = 2$)		$(n, 1024, 2d)$
Conv1D (Stride=4)	$(25, 2d, 4d)$	$(n, 256, 4d)$
LReLU ($\alpha = 0.2$)		$(n, 256, 4d)$
Phase Shuffle ($n = 2$)		$(n, 256, 4d)$
Conv1D (Stride=4)	$(25, 4d, 8d)$	$(n, 64, 8d)$
LReLU ($\alpha = 0.2$)		$(n, 64, 8d)$
Phase Shuffle ($n = 2$)		$(n, 64, 8d)$
Conv1D (Stride=4)	$(25, 8d, 16d)$	$(n, 16, 16d)$
LReLU ($\alpha = 0.2$)		$(n, 16, 16d)$
Reshape		$(n, 256d)$
Dense	$(256d, 1)$	$(n, 1)$

(a) WaveGAN Architecture([66])



(b) Loss-Curve



(c) Generated Signal After Amplification

Figure 4.18: WaveGan for Drone Audio Generation

class single drone in the scene. A sample from the generated audio for a duration of 1 second is shown in the figure 4.18(c). The generated audio samples can be found in [67]. However, due to the complexity of the multiple drone detection problem, a choice was made to perform the data augmentation by overlaying the drone detection data set with the Acoustic scene data set instead of using WaveGAN.

5 Augmenting Drone Detection Dataset

To comment on the performance of classifiers for multiple drone detection in real life cases, and the scalability of the system, the Drone Detection Dataset is been overlayed with the DCASE dataset with all the 10 acoustic scene classes. Each sample in the drone detection dataset is been overlayed with 10 samples from random classes of dcase dataset with 1 second duration. This resulted in a dataset with 14000 samples. Two sets of experiments are performed on augmented dataset, one with the architectures employed on original drone detection dataset, followed by deeper architectures. The anatomy of the augmented dataset corresponding to the 10 acoustic scenes of the dcase dataset is found in the table 5.1

Acoustic Scenes	Background Noise	Single Drone	Two Drone
Airport	308	757	310
Shopping Mall	333	827	298
Metro Station	271	736	290
Street Pedestrian	281	877	313
Public Square	309	819	292
Street Traffic	310	763	311
Tram	294	824	303
Bus	308	800	283
Metro	256	797	313
Park	330	800	287

Table 5.1: Anatomy of Augmented Drone Detection Dataset

6 Experiments: Drone Detection with Augmented Dataset

This section deals with the experiments performed on the problem of multiple drone detection with the augmented drone detection dataset. The experiments in this chapter are organized in the following order.

1) PCA and TSNE visualization of the SMILE988 features.

2) Random Forest Algorithm applied on the SMILE988 features.

3) Deep Neural Network with SMILE988 features.

4) Deep Neural Network with SMILE988 reduced features

5) Convolutional Neural Network with 3-channel spectrograms.

6) Convolutional Neural Network with 2- channel Spectrograms with Harmonic and Percussive content into individual channels.

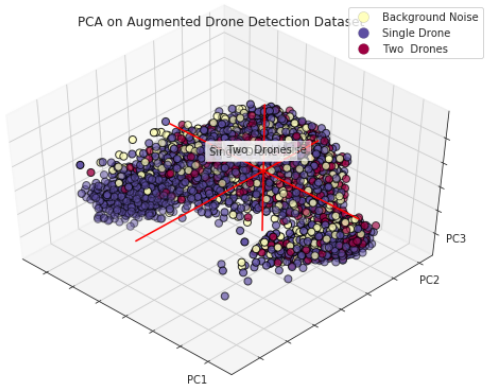
6.1 Experiment1: PCA and TSNE analysis

Similar to the section 4.1.1 of this document, the openSMILE large scale feature extraction is employed, extracting 988 features for each sample on the augmented drone detection dataset. In this experiment, the PCA and TSNE techniques are employed to visualize the

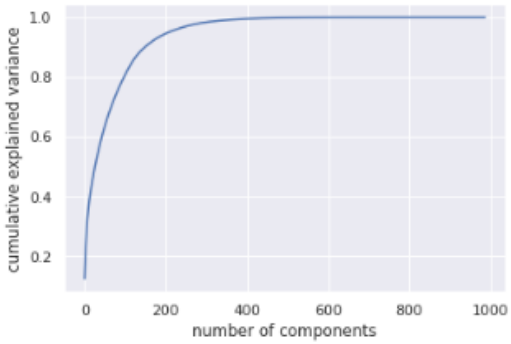
extracted features. The TSNE algorithm is trained for 3500 iterations with an early stopping mechanism in the absence of improvement over 300 iterations. The 15 most contributing features in computing the first three principal components are shown in the table 6.1.

Principal Component 1	Principal Component 2	Principal Component 3
F0_sma_linregerrQ	F0env_sma_linregerrQ	F0_sma_de_linregerrQ
F0env_sma_linregerrQ	F0_sma_linregerrQ	F0_sma_linregerrQ
F0_sma_de_linregerrQ	F0_sma_de_linregerrQ	F0env_sma_quartile1
F0env_sma_quartile2	F0_sma_max	F0env_sma_quartile2
F0env_sma_quartile3	F0_sma_range	F0env_sma_quartile3
F0env_sma_max	F0env_sma_range	F0env_sma_max
F0_sma_range	F0env_sma_linregc2	F0env_sma_range
F0_sma_max	F0env_sma_stddev	F0_sma_max
F0env_sma_range	F0env_sma_max	F0_sma_range
F0_sma_quartile3	F0env_sma_iqr1-3	F0env_sma_amean
F0env_sma_quartile1	F0_sma_iqr1-3	F0env_sma_linregc2
F0_sma_iqr1-3	F0_sma_quartile3	F0_sma_de_range
F0env_sma_amean	F0env_sma_quartile3	F0_sma_de_iqr1-3
F0env_sma_linregc2	F0_sma_de_range	F0_sma_de_min
F0_sma_iqr2-3	F0env_sma_de_linregerrQ	F0_sma_de_max

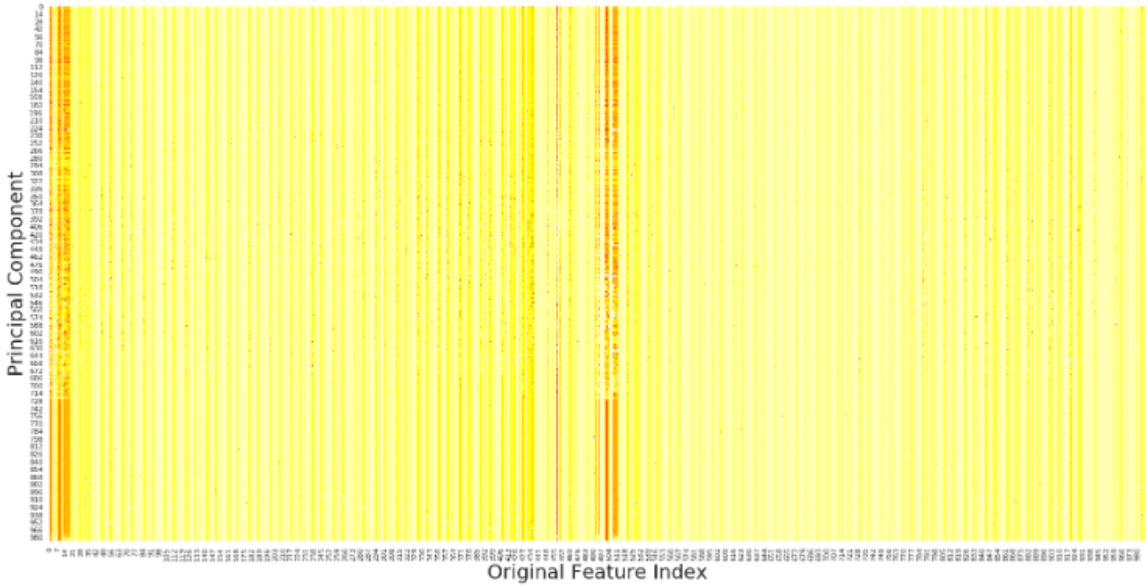
Table 6.1: Most Contributing Features for First 3-Principal Components for Augmented Drone Dataset



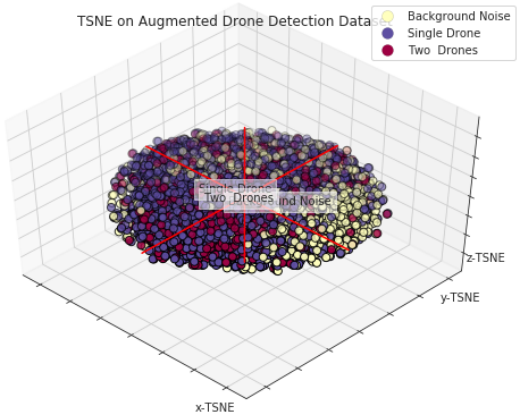
(a) PCA Visualization



(b) PCA Visualization



(c) PCA Visualization



(d) TSNE visualization

Figure 6.1: SMILE988 data visualization for augmented Drone detection dataset

6.2 Experiment 2: Random Forest Algorithm with SMILE 988 features

In PCA analysis, we have the most important features that contribute to the calculation of the first three-Principal components. However, PCA deals with the features as a group with a weighted average. To understand the information gain provided by the each individual feature in the dataset, random forest regressor is employed on the SMILE988 feature set. The regressor is trained with 2500 estimators, resulting in the classification accuracy of 63.3 percent. The top features set is populated by the mfcc's and linear predicative coefficients. The confusion matrix is shown in the figure 6.2 and 40 features with maximum information gain is shown in the figure 6.3.

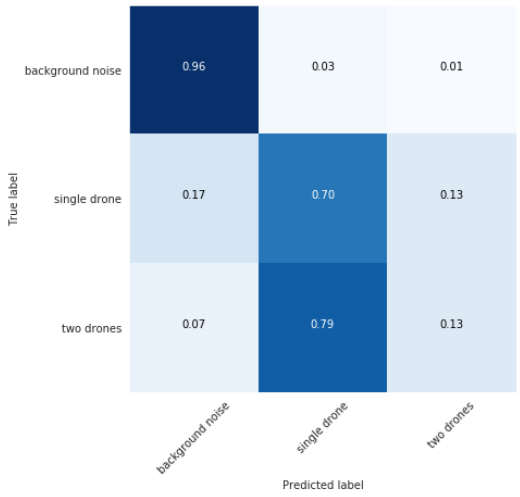
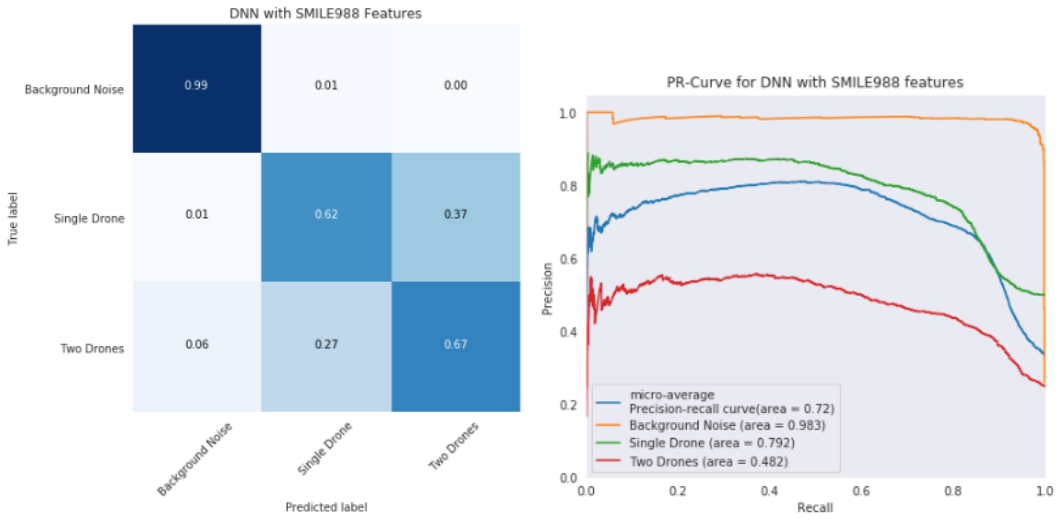


Figure 6.2: Confusion Matrix for Random Forest algorithm

Figure 6.3: 40 Most Contributing features for the Random Forest algorithm

6.3 Experiment 3: SMILE988 with DNN

The SMILE988 features extracted on the augmented drone detection dataset are fed into a five layer Deep Neural Network and was trained for 200 epochs. The hyper parameters involved in training the network include Categorical cross entropy loss function, Adam optimizer for Gradient Descent, Exponential Linear Units as activation functions and soft max activation in the final layer. This experiment resulted in classification accuracy of 76 percent, AUC for micro-average PR-curve is 0.72, and micro-average F1-score is 0.73.



(a) Confusion Matrix (b) PR-Curve

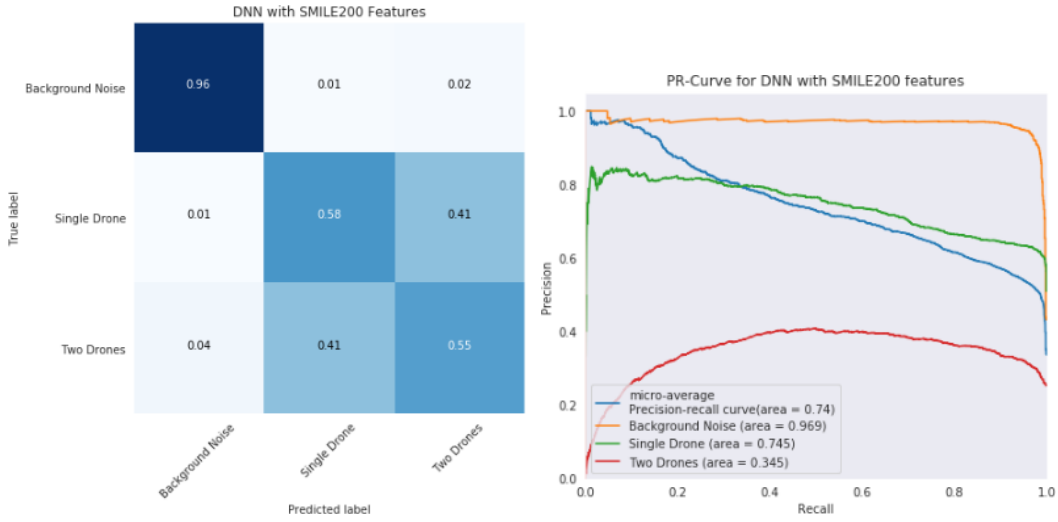
Figure 6.4: DNN with SMILE988 Results

Classes	Precision	Recall	F1-Score
Background Noise	0.93	0.99	0.96
Single Drone	0.82	0.62	0.71
Two Drones	0.48	0.67	0.56
avg/total	0.76	0.73	0.73

Table 6.2: Classification Report for DNN with SMILE988 Features

6.4 Experiment 4: DNN with SMILE200 features

In general, only a subset of features in a dataset actually contribute in increasing the performance of the classifier, the rest just add noise in the environment and effect the performance of the classifier. The subset of features contributing in calculation of first three principal components are extracted and fed into a DNN. The network is trained for 200 epochs with a batch size of 128. This experiment resulted in the classification accuracy of 69 percent. Since, the dataset is augmented by overlaying two different distributions, variance contributed by individual feature is decreased. This experiment resulted in classification accuracy of 69 percent, AUC for micro-average PR-curve is 0.74, and micro-average F1-score is 0.68.



(a) Confusion Matrix (b) PR-Curve

Figure 6.5: DNN with SMILE200 Results

Classes	Precision	Recall	F1-Score
Background Noise	0.94	0.96	0.95
Single Drone	0.74	0.58	0.65
Two Drones	0.40	0.55	0.46
avg/total	0.70	0.67	0.68

Table 6.3: Classification Report for DNN with SMILE200 Features

6.5 Experiment 5: CNN with Spectrograms

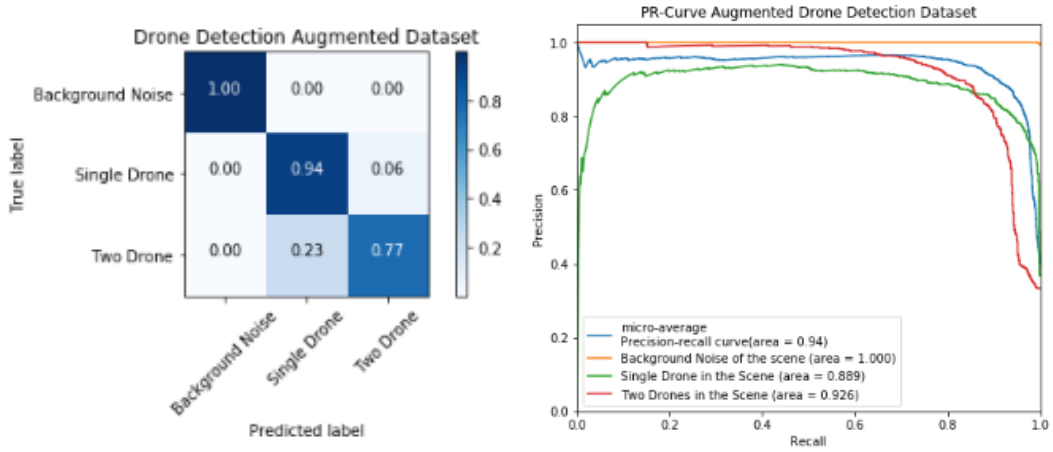
The augmented drone detection dataset consists of a 14000 audio samples, the 10 fold increase in the dataset size enabled to utilize deeper architectures. The CNN employed in this experiment has eight convolutional layers. However, the test environment is similar to the one in the section 4.1.2 of this document. In the series of experiments, Variants of spectro-temporal features including spectrograms, log-spectrograms, melspectrograms, log-mel spectrograms with 40, 60, 80, 128, 200 filters are performed.

The three channel raw spectrograms resulted by applying the stft on the left, right, left - right channels of the stereo audio, with a window size of 2048 samples and overlap of 512 samples are fed into the 8-Conv layer CNN. The network is trained to minimize the categorical cross entropy loss, rmsprop optimization and 'elu' activations. The model was trained for 200 epochs with a batch size of 16 samples.

The deeper architectures has resulted in consistent and better classification accuracy. This experiment resulted in classification accuracy of 90.3 percent, AUC for micro-average PR-curve is 0.94, and micro-average F1-score is 0.90.

6.5.1 CNN with log spectrograms

The spectrograms extracted in the experiment 3 are transformed into logarithmic domain. The transformed features are used as the input to the model. The network is trained for 200 epochs with batch size of 16, minimizing the categorical cross-entropy loss, RmsProp optimization and Exponential Linear Units activation functions.This experiment resulted in classification accuracy of 91 percent, AUC for micro-average PR-curve is 0.96, and micro-average F1-score is 0.91.



(a) Confusion Matrix (b) PR-Curve

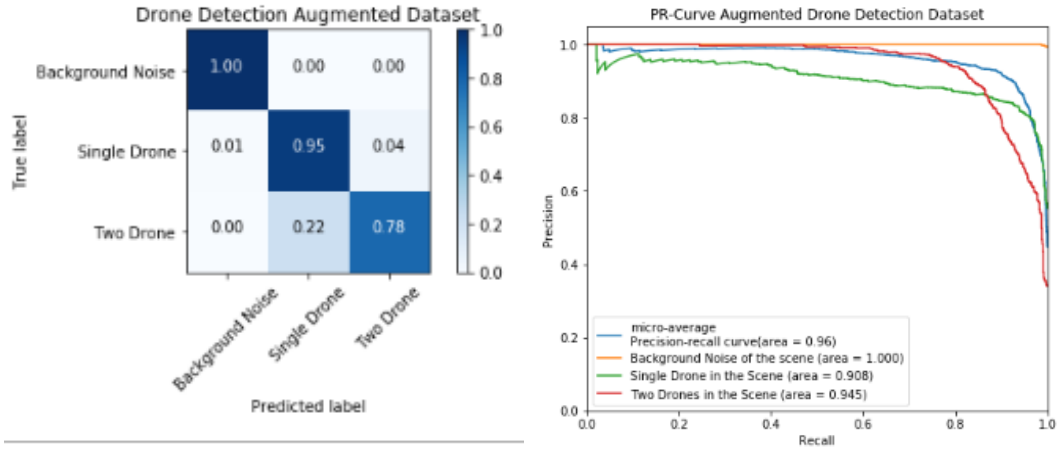
Figure 6.6: CNN with raw Spectrograms Results

Classes	Precision	Recall	F1-Score
Background Noise	1	1	1
Single Drone	0.80	0.94	0.87
Two Drones	0.93	0.77	0.84
avg/total	0.91	0.90	0.90

Table 6.4: Classification Report for CNN with raw Spectrograms

6.5.2 CNN with MelSpectrograms with 128 Mel filters

For the 3-channel Spectrograms extracted in the experiment 3. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 73.6 percent, AUC for micro-average PR-curve is 0.85, and micro-average F1-score is 0.73.



(a) Confusion Matrix (b) PR-Curve

Figure 6.7: CNN with Log-Spectrograms Results

Classes	Precision	Recall	F1-Score
Background Noise	0.99	1	1
Single Drone	0.81	0.95	0.88
Two Drones	0.95	0.78	0.86
avg/total	0.92	0.91	0.91

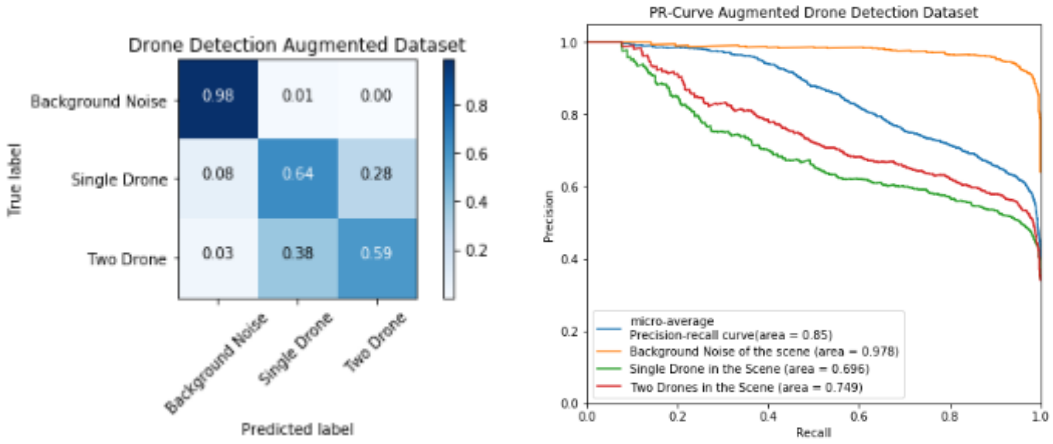
Table 6.5: Classification Report for CNN with Log-Spectrograms

6.5.3 CNN with Log-Mel Spectrograms with 40 Mels

For the 3-channel Spectrograms extracted in the previous experiment. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 85.6 percent, AUC for micro-average PR-curve is 0.95, and micro-average F1-score is 0.86.

6.5.4 CNN with Log-Mel Spectrograms with 60 Mels

For the 3-channel Spectrograms extracted in the experiment 3. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is



(a) Confusion Matrix (b) PR-Curve

Figure 6.8: CNN with Melspectrograms with 128 Mel filters Results

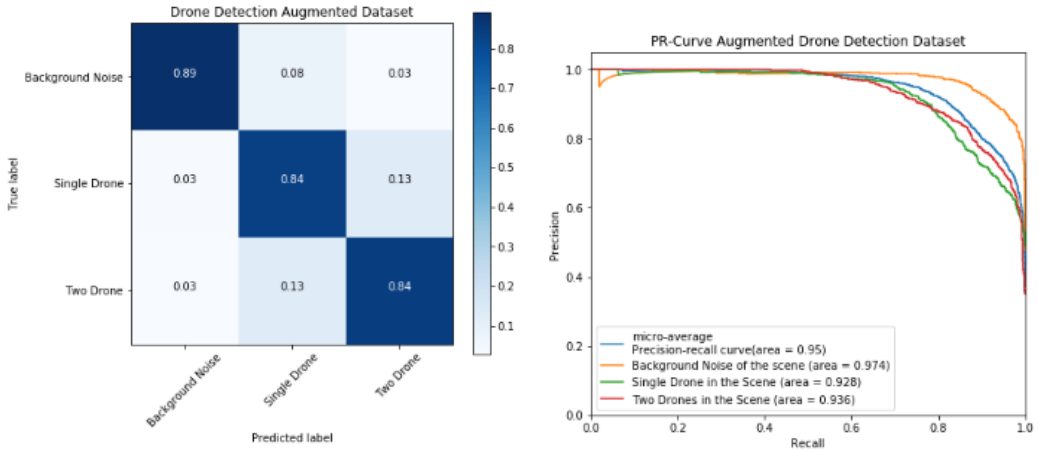
Classes	Precision	Recall	F1-Score
Background Noise	0.89	0.98	0.94
Single Drone	0.62	0.64	0.63
Two Drones	0.67	0.59	0.63
avg/total	0.73	0.74	0.73

Table 6.6: Classification Report for CNN with Mel-Spectrograms with 128 Mels

trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 87 percent, AUC for micro-average PR-curve is 0.95, and micro-average F1-score is 0.87.

6.5.5 CNN with Log-Mel Spectrograms with 80 Mels

For the 3-channel Spectrograms extracted in the experiment 3. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 87 percent, AUC for micro-average PR-curve is 0.95,



(a) Confusion Matrix (b) PR-Curve

Figure 6.9: CNN with Log-MelSpectrograms with 40 Mels Results

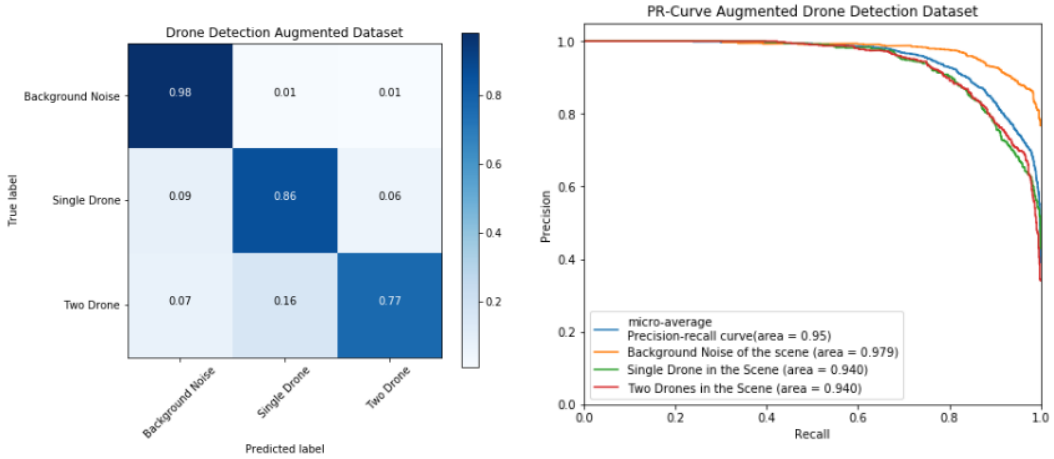
Classes	Precision	Recall	F1-Score
Background Noise	0.93	0.89	0.91
Single Drone	0.80	0.84	0.82
Two Drones	0.84	0.84	0.84
avg/total	0.86	0.86	0.86

Table 6.7: Classification Report for CNN with Log-Mel Spectrograms with 40 Mels

and micro-average F1-score is 0.86.

6.5.6 CNN with Log-Mel Spectrograms with 128 Mels

For the 3-channel Spectrograms extracted in the experiment 3. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 85.3 percent, AUC for micro-average PR-curve is 0.95, and micro-average F1-score is 0.86.



(a) Confusion Matrix (b) PR-Curve

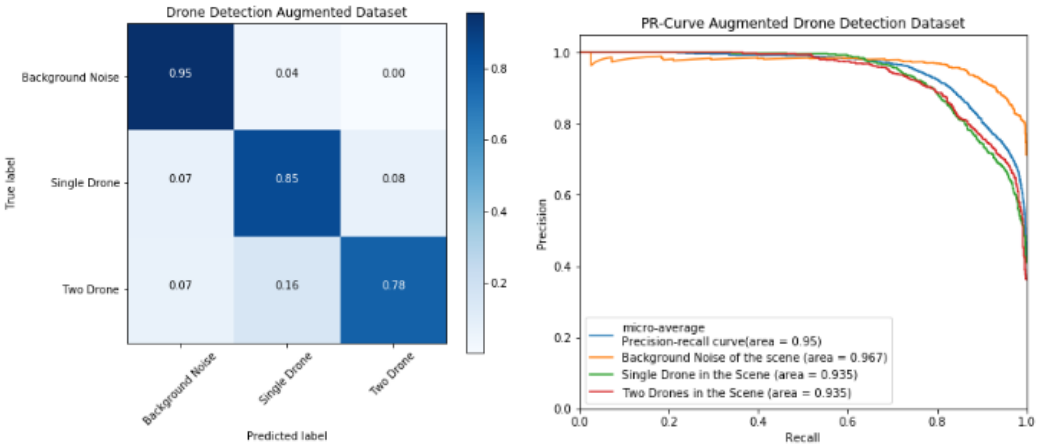
Figure 6.10: CNN with Log-MelSpectrograms with 60 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.86	0.98	0.91
Single Drone	0.83	0.86	0.84
Two Drones	0.92	0.77	0.84
avg/total	0.87	0.87	0.87

Table 6.8: Classification Report for CNN with Log-Mel Spectrograms with 60 Mels

6.5.7 CNN with Log-Mel Spectrograms with 200 Mels

For the 3-channel Spectrograms extracted in the experiment 3. The Mel filter bank with 128 Mel filters is applied, resulting in a three-channel Mel-Spectrogram. The 8-layer CNN is trained for 200 epochs with batch size of 128. The training is performed using the RmsProp optimizer algorithm and minimizing the categorical cross-entropy loss. This experiment resulted in classification accuracy of 87 percent, AUC for micro-average PR-curve is 0.95, and micro-average F1-score is 0.87.

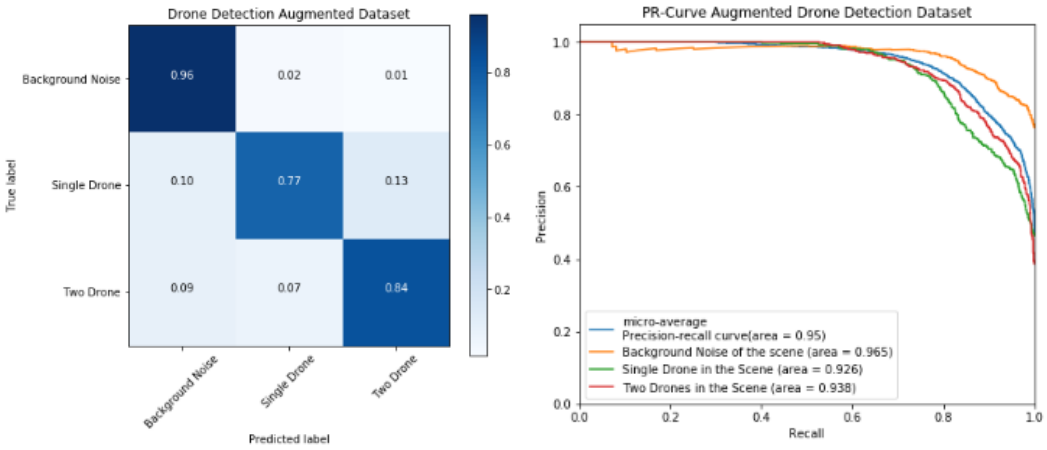


(a) Confusion Matrix (b) PR-Curve

Figure 6.11: CNN with Log-MelSpectrograms with 80 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.87	0.95	0.91
Single Drone	0.81	0.85	0.83
Two Drones	0.91	0.78	0.84
avg/total	0.86	0.86	0.86

Table 6.9: Classification Report for CNN with Log-Mel Spectrograms with 80 Mels

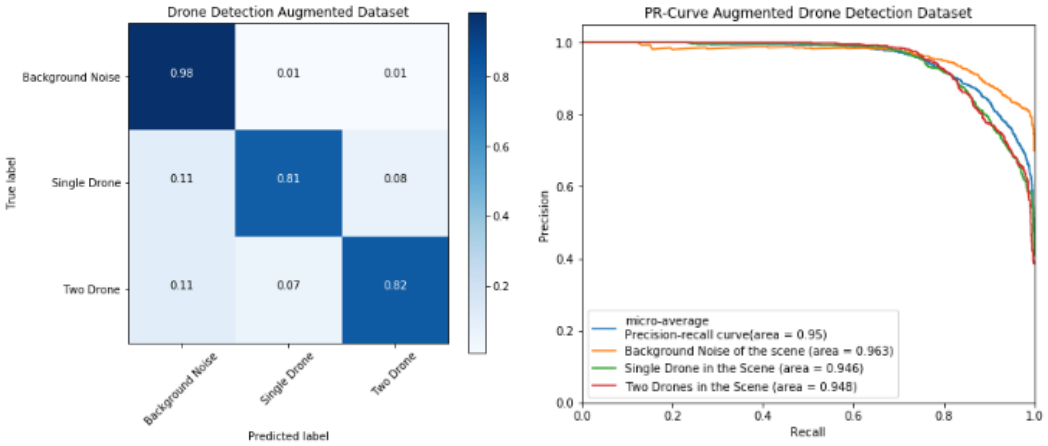


(a) Confusion Matrix (b) PR-Curve

Figure 6.12: CNN with Log-MelSpectrograms with 128 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.84	0.96	0.90
Single Drone	0.89	0.77	0.83
Two Drones	0.90	0.84	0.84
avg/total	0.86	0.86	0.86

Table 6.10: Classification Report for CNN with Log-Mel Spectrograms with 128 Mels



(a) Confusion Matrix (b) PR-Curve

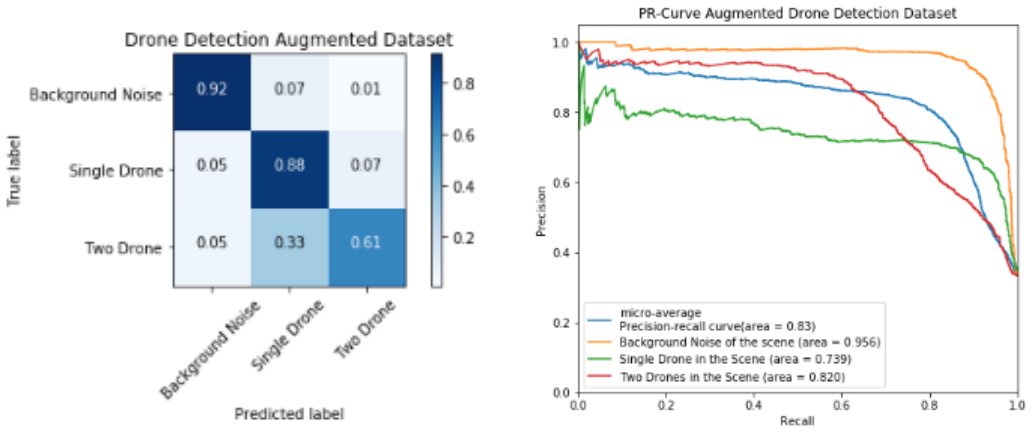
Figure 6.13: CNN with Log-MelSpectrograms with 200 Mels Results

Classes	Precision	Recall	F1-Score
Background Noise	0.81	0.98	0.89
Single Drone	0.91	0.81	0.86
Two Drones	0.90	0.82	0.86
avg/total	0.88	0.87	0.87

Table 6.11: Classification Report for CNN with Log-Mel Spectrograms with 200 Mels

6.6 Experiment 6: Harmonic Percussive Source Separation

In this experiment, the stereo audio is converted into monophonic audio and STFT is computed on it. The harmonic and the percussive content of the spectrogram is separated into two individual channels. The two-channel hpss spectrograms are fed into the cnn. The model is trained for 200 epochs with a batch size of 16, the network is trained to minimize categorical cross-entropy is used to measure the loss, and adam optimization.This experiment resulted in classification accuracy of 81 percent, AUC for micro-average PR-curve is 0.83, and micro-average F1-score is 0.80.



(a) Confusion Matrix (b) PR-Curve

Figure 6.14: CNN with Harmonic Percussive Source Separation Results

Classes	Precision	Recall	F1-Score
Background Noise	0.90	0.92	0.91
Single Drone	0.68	0.88	0.77
Two Drones	0.88	0.61	0.72
avg/total	0.82	0.80	0.80

Table 6.12: Classification Report for CNN with Harmonic Percussive Source Separation

7 Experiments: DCASE

7.1 Baseline System

7.1.1 Baseline System

The two-layer convolutional neural network is provided as the baseline system by the DCASE2018 organizers. The reported class wise accuracy on the baseline system is shown in table 7.1. Two-channel normalized Log-Mel Spectrograms corresponding to the two channels of the stereo audio are fed into the CNN for training the baseline system. The Log-Mel Spectrograms are extracted with 40 Mel filters on the frame size of 1928 samples and hop length of 500 samples. The hyper-parameters of the baseline system include the categorical cross entropy loss function, categorical accuracy as the performance measure, and Adam optimization algorithm. The model was trained for 200 epochs with 16 samples per batch yielding the final validation accuracy of 59.7 percent. The architecture of the baseline system was shown in figure 7.1.

Scene Class	Accuracy
Airport	72.9%
Bus	62.9%
Metro	51.2%
Metro Station	55.4%
Park	79.1%
Public Square	40.4%
Shopping Mall	49.6%
Street,Pedestrian	50.0%
Street,Traffic	80.5%
Tram	55.1%
Average	59.7%

Table 7.1: Class-wise Performance of DCASE Baseline system

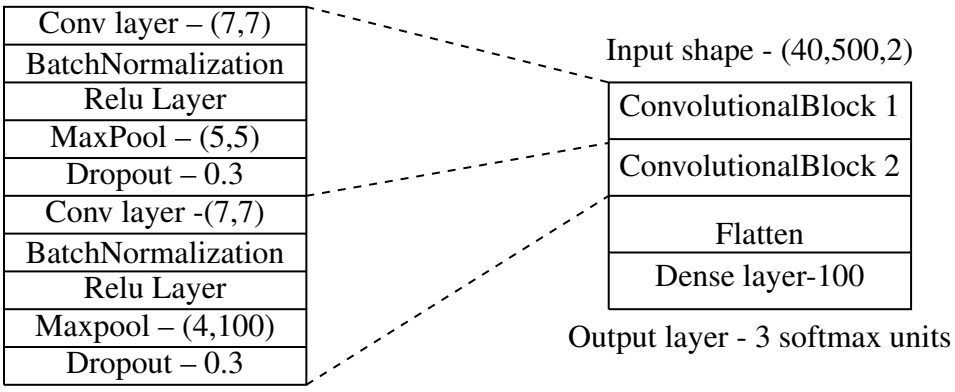
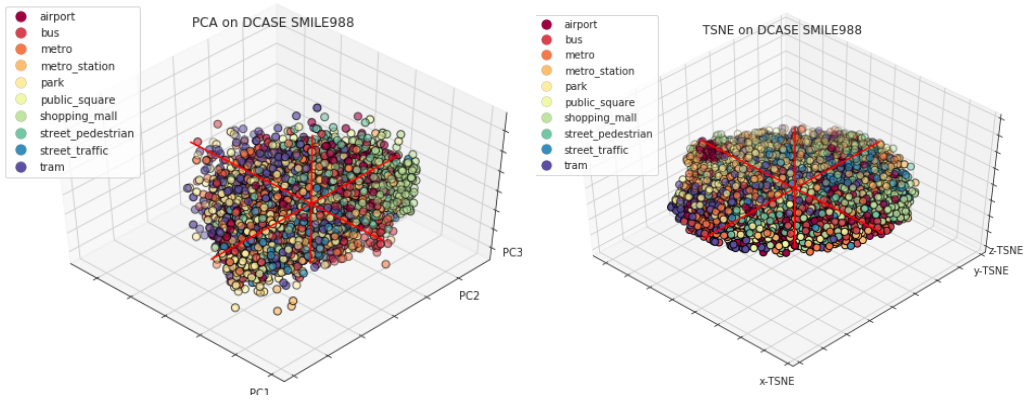


Figure 7.1: Baseline System Architectue

7.2 PCA and TSNE visualization for DCASE SMILE988 Features

SMILE988 features are extracted for each audio sample in the DCASE dataset. Since the dataset exists in a very high-dimensional space, the linear and non-linear dimensionality reduction techniques like Principal component analysis(PCA) and t-distributed Stochastic Neighbor Embedding(TSNE) are used to reduce the data into 3-dimensional space. Dimensionality reduction is performed to visualize the data and to observe the variance in the dataset. The three-dimensional visualization of the PCA and TSNE results are shown in figure 7.2(a) and figure 7.2(b) respectively. The set of top 15 most contributed features in computing the first three principal components are shown in table 7.2.



(a) PCA (b) TSNE

Figure 7.2: Data Visualization in reduced dimensions for DCASE SMILE 988

Principal Component 1	Principal Component 2	Principal Component 3
F0_sma_linregerrQ	F0env_sma_linregerrQ	lspFreq_sma[0]_maxPos
F0_sma_de_linregerrQ	F0_sma_linregerrQ	mfcc_sma[1]_minPos
F0env_sma_linregerrQ	F0env_sma_de_minPos	pcm_zcr_sma_maxPos
F0_sma_quartile3	F0env_sma_de_maxPos	pcm_zcr_sma_de_maxPos
F0_sma_iqr1-3	voiceProb_sma_de_maxPos	lspFreq_sma_de[1]_maxPos
F0_sma_iqr2-3	F0env_sma_linregc2	lspFreq_sma[1]_maxPos
F0env_sma_linregc2	voiceProb_sma_maxPos	lspFreq_sma_de[1]_minPos
F0_sma_amean	F0env_sma_stddev	lspFreq_sma_de[0]_maxPos
F0_sma_linregc2	F0_sma_de_linregerrQ	pcm_zcr_sma_de_minPos
F0env_sma_de_maxPos	F0env_sma_range	mfcc_sma_de[1]_minPos
F0env_sma_quartile1	F0_sma_range	lspFreq_sma_de[0]_minPos
F0_sma_linregerrA	F0_sma_max	lspFreq_sma[2]_maxPos
F0env_sma_maxPos	F0env_sma_max	lspFreq_sma_de[2]_maxPos
F0_sma_stddev	F0env_sma_linregerrA	mfcc_sma_de[1]_maxPos
F0env_sma_amean	F0env_sma_quartile3	lspFreq_sma_de[2]_minPos

Table 7.2: Most Contributing Features for First 3-Principal Components for DCASE SMILE988

7.3 Random Forest Algorithm on DCASE SMILE988 Features

To understand the information gain provided by each individual feature for final prediction of class, random forest is used. The random forest regressor is trained with 1000 estimator, resulting in the total classification accuracy of 57 percent. The confusion matrix for this experiment is shown in figure 7.4, and the top 30 features providing the maximum information gain is shown in figure 7.3.

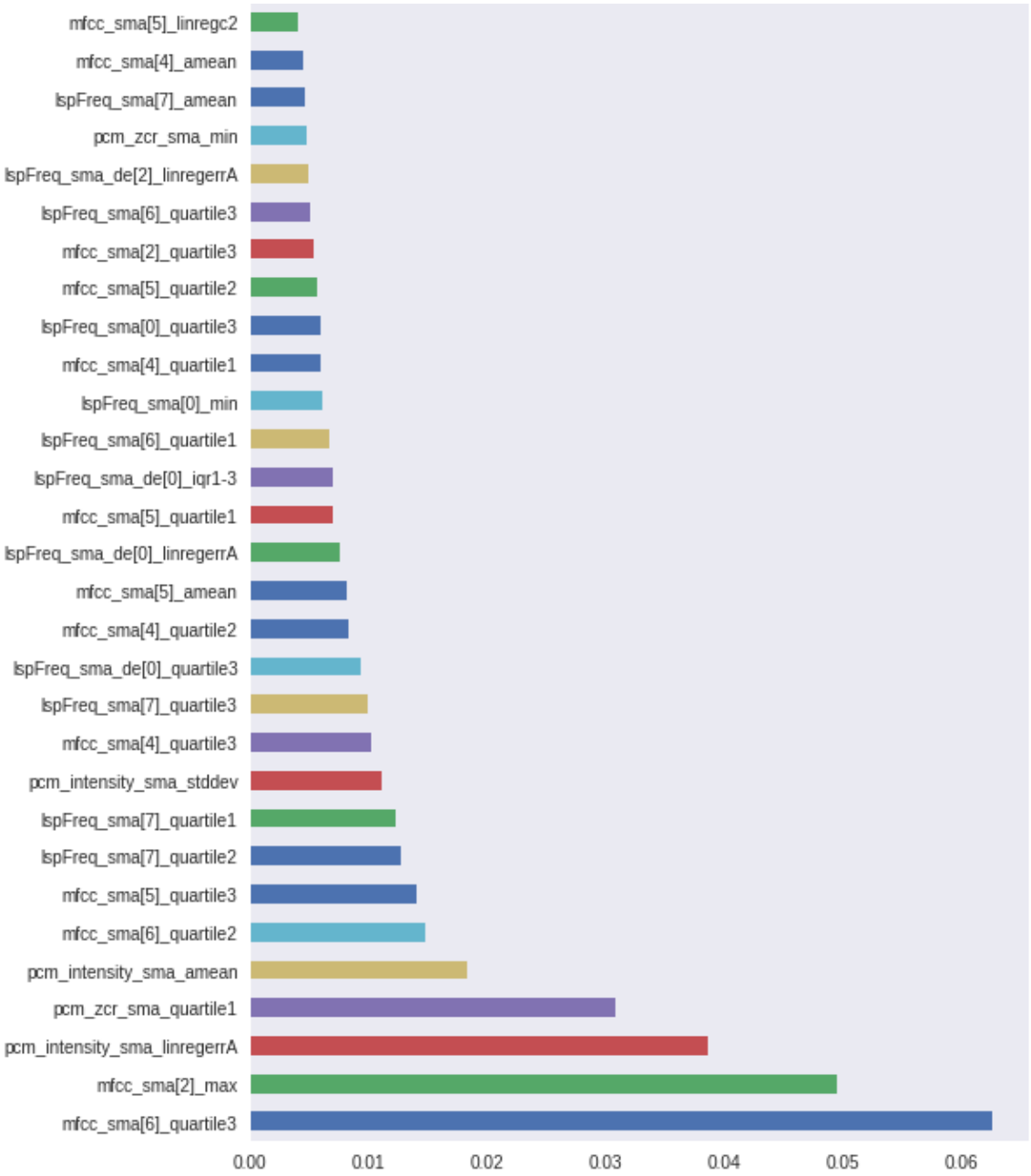
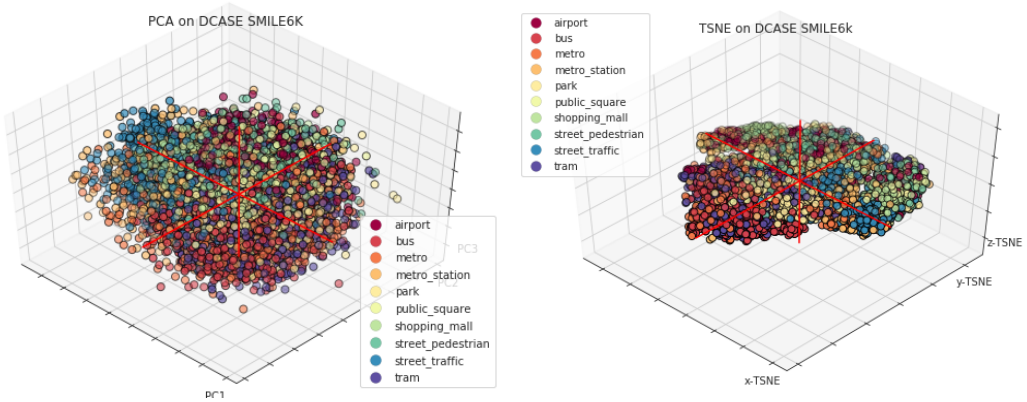


Figure 7.3: 30 Most Contributing features for the Random Forest algorithm

Figure 7.4: Confusion Matrix for the Random Forest algorithm

7.4 PCA and TSNE visualization for DCASE SMILE6k Features

Since the previous experiment, the random forest algorithm with SMILE988 features yielded a total classification accuracy below the baseline. So, in this experiment, [59] configuration file is employed to extract all the extractable features from the openSMILE library. A total of 6552 features are extracted for each audio sample in the DCASE dataset. The results of the PCA and TSNE in three-dimensional space is shown in figure 7.5. The most contributing features for calculating the first three principal components in PCA are shown in table 7.3.



(a) PCA (b) TSNE

Figure 7.5: Data Visualization in reduced dimensions for SMILE6k

Principal Component 1	Principal Component 2	Principal Component 3
pcm_fftMag _melspec_sma[20] _qregerrQ	pcm_fftMag _melspec_sma[23] _qregerrQ	pcm_fftMag _melspec_sma[12] _qregerrQ
pcm_fftMag _melspec_sma[25] _qregerrQ	pcm_fftMag _melspec_sma[24] _qregerrQ	pcm_fftMag _melspec_sma[11] _qregerrQ
pcm_fftMag _melspec_sma[19] _qregerrQ	pcm_fftMag _melspec_sma[19] _qregerrQ	pcm_fftMag _melspec_sma[15] _qregerrQ
pcm_fftMag _melspec_sma[21] _qregerrQ	pcm_fftMag _melspec_sma[22] _qregerrQ	pcm_fftMag _melspec_sma[17] _qregerrQ
pcm_fftMag _melspec_sma[24] _qregerrQ	pcm_fftMag _melspec_sma[20] _qregerrQ	pcm_fftMag _melspec_sma[13] _qregerrQ
pcm_fftMag _melspec_sma[18] _qregerrQ	pcm_fftMag _melspec_sma[21] _qregerrQ	pcm_fftMag _melspec_sma[16] _qregerrQ
pcm_fftMag _melspec_sma[23] _qregerrQ	pcm_fftMag _melspec_sma[25] _qregerrQ	pcm_fftMag _melspec_sma[18] _qregerrQ
pcm_fftMag _melspec_sma[22] _qregerrQ	pcm_fftMag _melspec_sma[18] _qregerrQ	pcm_fftMag _melspec_sma[14] _qregerrQ
pcm_fftMag _melspec_sma[17] _qregerrQ	pcm_fftMag _melspec_sma[15] _qregerrQ	pcm_fftMag _melspec_sma[10] _qregerrQ
pcm_fftMag _melspec_sma[16] _qregerrQ	pcm_fftMag _melspec_sma[14] _qregerrQ	pcm_fftMag _melspec_sma[9] _qregerrQ
pcm_fftMag _melspec_sma[15] _qregerrQ	pcm_fftMag _melspec_sma[16] _qregerrQ	pcm_fftMag _melspec_sma[25] _qregerrQ
pcm_fftMag _melspec_sma[14] _qregerrQ	pcm_fftMag _melspec_sma[17] _qregerrQ	pcm_fftMag _melspec_sma[8] _qregerrQ
pcm_fftMag _melspec_sma[13] _qregerrQ	pcm_fftMag _melspec_sma[13] _qregerrQ	pcm_fftMag _melspec_sma[20] _qregerrQ
pcm_fftMag _melspec_sma[12] _qregerrQ	pcm_fftMag _melspec_sma[12] _qregerrQ	pcm_fftMag _melspec_sma[23] _qregerrQ
pcm_fftMag _melspec_sma_de[20] _qreg errQ	pcm_fftMag _melspec_sma[11] _qregerrQ	pcm_fftMag _melspec_sma[22] _qregerrQ

Table 7.3: Most Contributing Features for First 3-Principal Components for DCASE SMILE6K

7.5 Random Forest algorithm for DCASE SMILE6K features

Similar to section 7.3 of this document, the Random Forest algorithm is employed on the extracted SMILE6K features to understand the information gain provided by each of the individual features in predicting the final mean prediction. The random forest regressor is trained with 1000 estimators resulting in the final classification accuracy of 64 percent, which is 4.3 percent above the baseline system provided. The confusion matrix between the final mean predictions and the actual labels for the validation set is shown in figure 7.7. Figure 7.6 consists of the bar plot for the 40 features providing the maximum information gain.

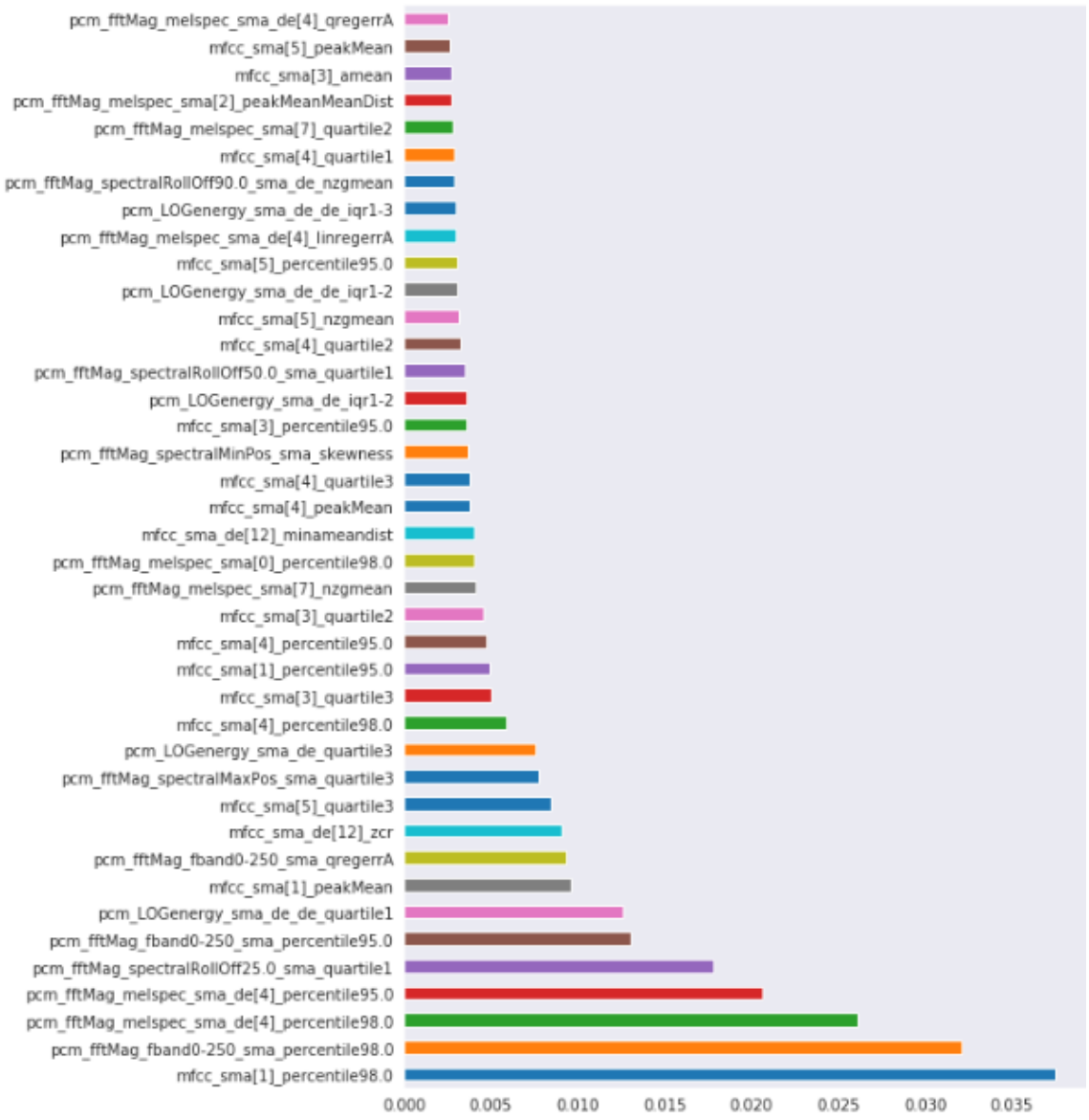


Figure 7.6: 40 Most Contributing features for the Random Forest algorithm

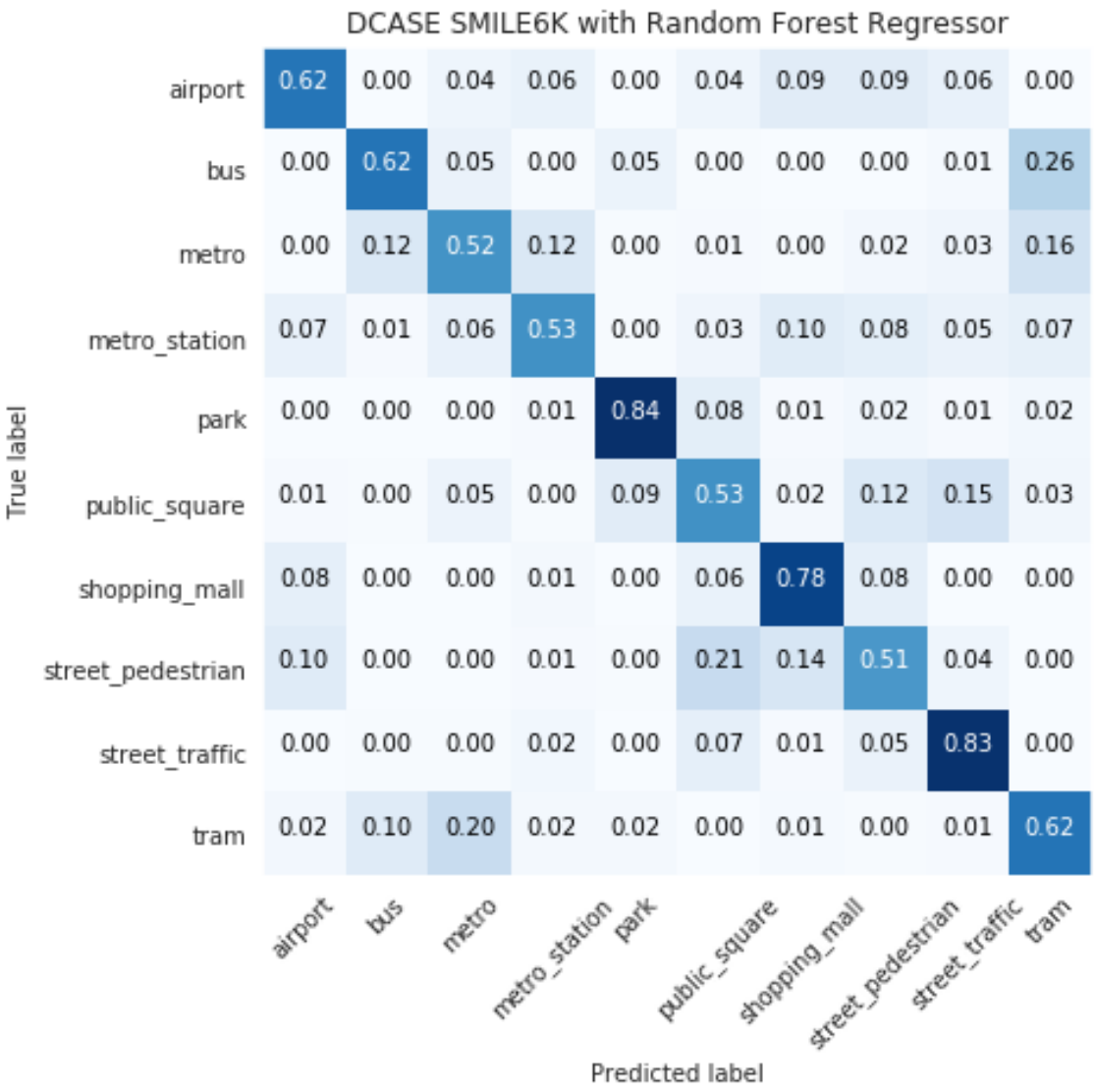


Figure 7.7: Confusion Matrix for the Random Forest algorithm

7.6 DNN with SMILE988 features

In this experiment, the 5-layer Deep Neural Network is employed with the smile988 feature set. The network is trained for over 800 epochs with a batch size of 128. The categorical accuracy loss function with Adam optimizer is employed. The three-layer hierarchical DNN shown in [68] has resulted in the state of the performance on DCASE dataset used for 2016 challenge. The experimental trails did begin with this architecture, with 512 neurons in each layer. It is observed that increasing the number of neurons in each layer degrades the generalization performance. However, increasing the depth of the model improved the performance of the model. The best performing model resulted in a classification accuracy of 65.3 percent. The architecture for the model used is shown in figure 7.8 and the prediction results are shown in the figure 7.9.

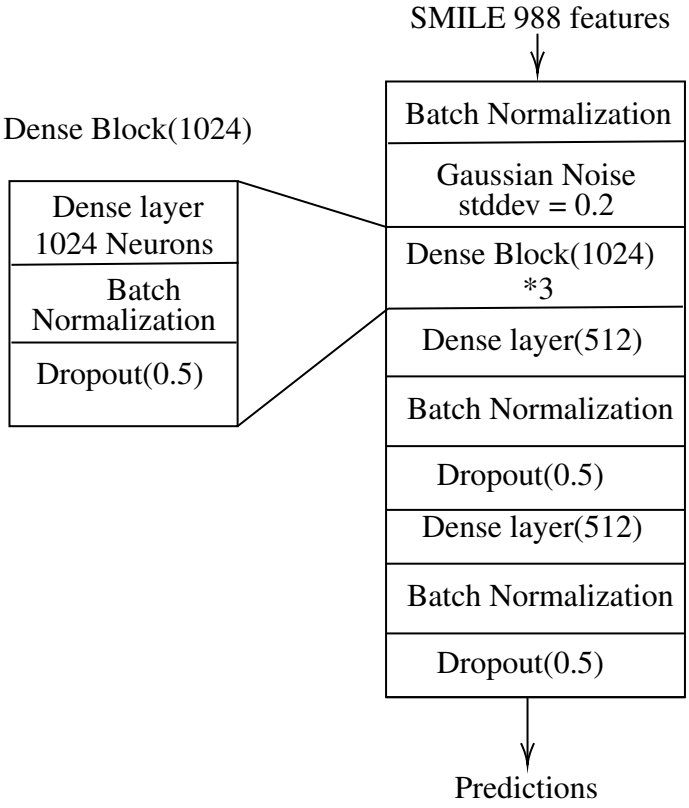
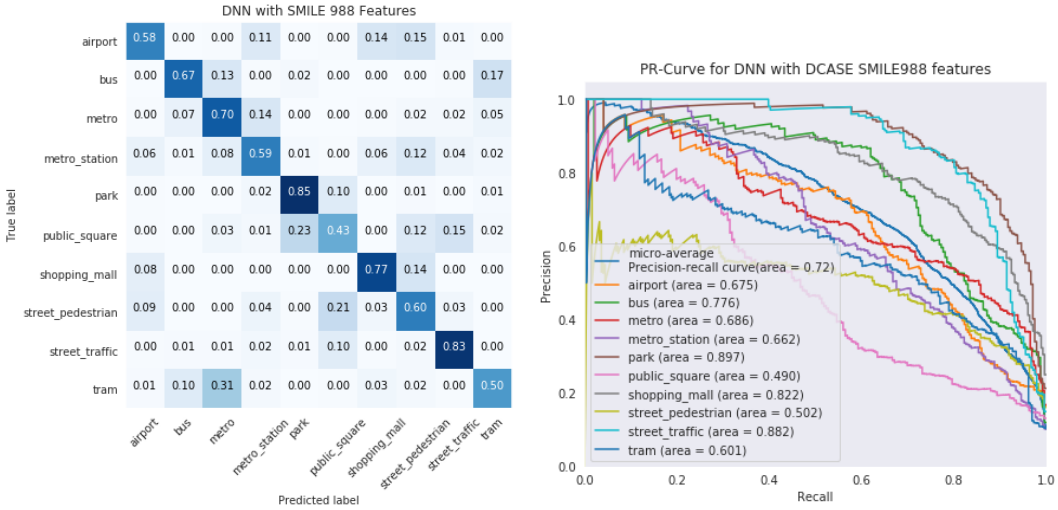


Figure 7.8: DNN architecture with SMILE features



(a) Confusion Matrix (b) PR-Curve

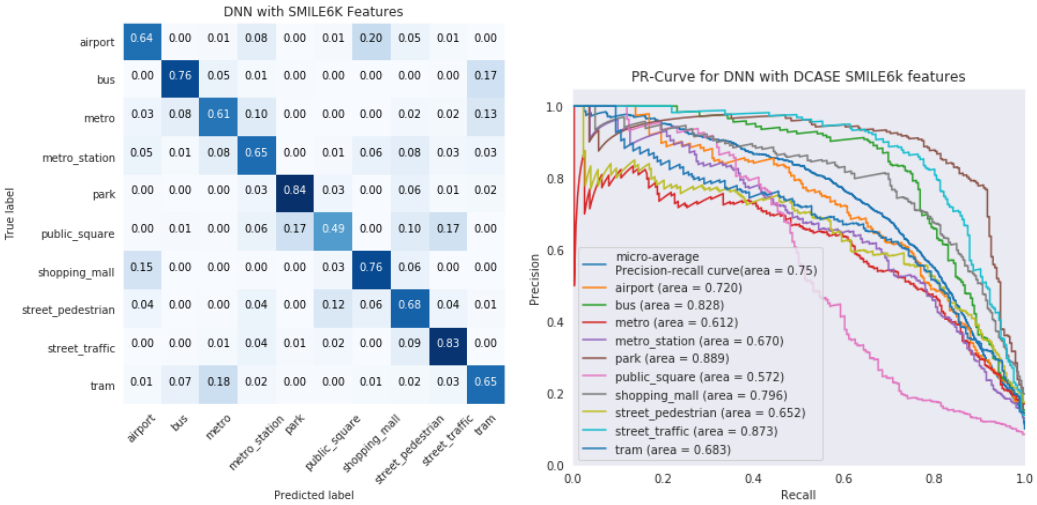
Figure 7.9: DNN with DCASE SMILE988 features on DCASE

Classes	Precision	Recall	F1-score	Support
airport	0.70	0.58	0.64	265
bus	0.77	0.67	0.72	242
metro	0.56	0.70	0.63	261
metro station	0.61	0.59	0.60	259
park	0.77	0.85	0.81	242
public square	0.47	0.43	0.45	216
shopping mall	0.76	0.77	0.77	279
street pedestrian	0.49	0.60	0.54	247
street traffic	0.77	0.83	0.80	246
tram	0.66	0.50	0.57	261
avg/total	0.66	0.66	0.65	2518

Table 7.4: Classification report DNN with DCASE SMILE988 features

7.7 DNN with SMILE6K features

In this experiment the 5-layer DNN shown in figure 7.8 is trained with SMILE6k features for 800, with categorical cross-entropy as loss function and batch size of 1024. This experiment has shown a significant improvement in the classification accuracy with 9.5 percent over the baseline model. The results from this experiment are shown in figure 7.10



(a) Confusion Matrix (b) PR-Curve

Figure 7.10: DNN with DCASE SMILE6k features on DCASE

Classes	Precision	Recall	F1-score	Support
airport	0.68	0.64	0.66	265
bus	0.80	0.76	0.78	242
metro	0.64	0.61	0.63	261
metro station	0.63	0.65	0.64	259
park	0.83	0.84	0.84	242
public square	0.65	0.49	0.56	216
shopping mall	0.71	0.76	0.73	279
street pedestrian	0.59	0.68	0.63	247
street traffic	0.74	0.83	0.78	246
tram	0.66	0.65	0.65	261
avg/total	0.69	0.69	0.69	2518

Table 7.5: Classification report DNN with DCASE SMILE6k features

7.8 CNN: Extended Baseline Model with data augmentation

While training the baseline model, it is observed that a significant amount of over-fitting. Extending the model for better architectures had resulted in degraded generalization performance. To improve the classification accuracy of the model, the DCASE dataset is augmented such that, each audio sample of 10sec duration is split into 20 samples with each of half second duration. The Log-Mel spectrograms are extracted with 128 Mel filters with a window size of 2048 samples and overlap of 512 samples. The network is trained for 200 epochs with adam optimization. During validation, the final label of the audio sample is predicted by taking the mean of the predictions obtained on the augmented samples corresponding to the original. This experiment resulted in the final accuracy of 68.5 percent which is 8.8 percent improvement over the base line model. The model architecture is shown in the figure 7.11.

7.8.1 CNN with RawAudio waveforms

In this experiment, the sample-level CNN architecture in [69] is used on the raw audio waveforms. The 2-channel raw audio waveform is sampled at a frequency of 22050 Hz. In each epoch, random slice from the original audio samples with 60000 samples is fed into the network. This network was trained for over 1000 epochs with Adam optimizer and a batch size of 4. During validation five overlapping frames with 60000 samples for the frame is fed into the network for every sample. The final label is predicted from the mean of the predictions on this intermediate frames. The final classification accuracy is 61.3 percent which is 0.6 percent over the baseline performance. The limited number of experiments are performed on this model, since any minor change in the architecture is

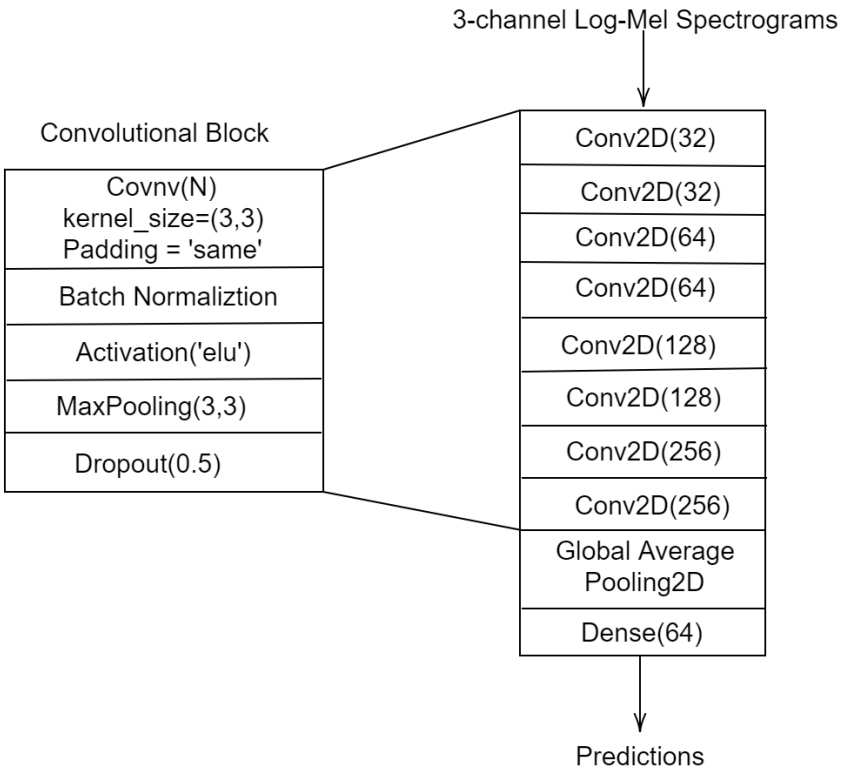
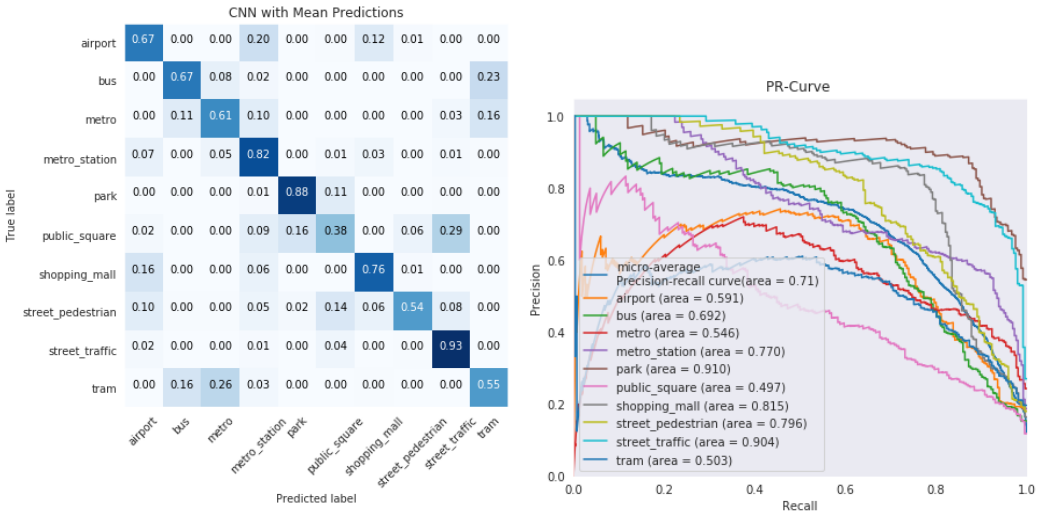


Figure 7.11: Extended Baseline Model



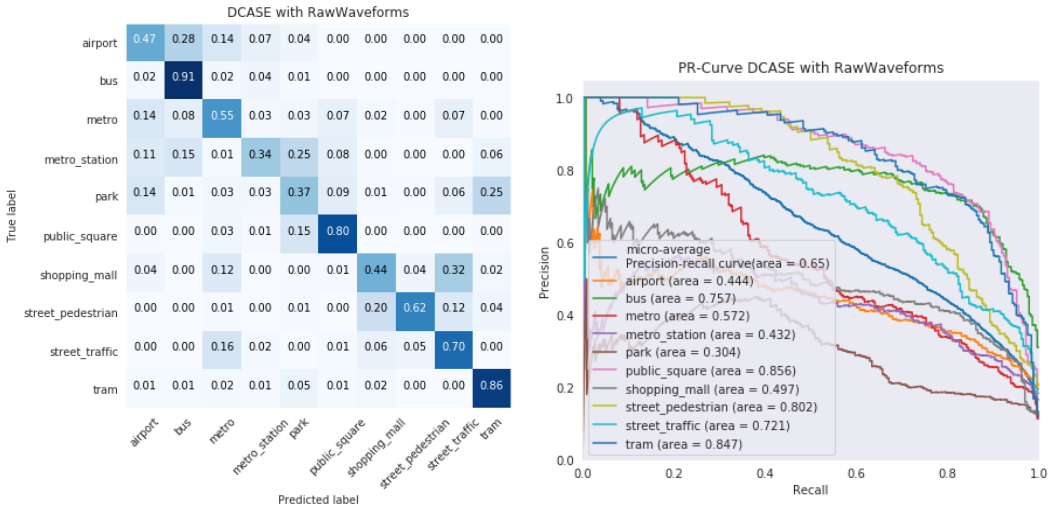
(a) Confusion Matrix (b) PR-Curve

Figure 7.12: CNN with DCASE Extended baseline model and data augmentation

Classes	Precision	Recall	F1-score	Support
airport	0.65	0.67	0.66	265
bus	0.69	0.67	0.68	242
metro	0.61	0.61	0.61	261
metro station	0.59	0.82	0.69	259
park	0.85	0.88	0.86	242
public square	0.53	0.38	0.44	216
shopping mall	0.79	0.76	0.77	279
street pedestrian	0.88	0.54	0.67	247
street traffic	0.71	0.93	0.80	246
tram	0.60	0.55	0.57	261
avg/total	0.69	0.68	0.68	2518

Table 7.6: Classification report for Extended baseline CNN with data augmentation

resulting from an inferior performance, and computationally expensive.



(a) Confusion Matrix (b) PR-Curve

Figure 7.13: CNN with raw audio waveforms

Classes	Precision	Recall	F1-score	Support
airport	0.52	0.47	0.50	265
bus	0.65	0.91	0.76	242
metro	0.50	0.55	0.53	261
metro station	0.63	0.34	0.44	259
park	0.37	0.37	0.37	242
public square	0.75	0.80	0.78	216
shopping mall	0.60	0.44	0.51	279
street pedestrian	0.86	0.62	0.72	247
street traffic	0.56	0.70	0.62	246
tram	0.71	0.86	0.78	261
avg/total	0.62	0.61	0.60	2518

Table 7.7: Classification report for CNN with raw waveforms

Hierarchical ensemble of Convolutional Neural Networks

After the careful examination of the results in the previous experiments, a structured pattern of mutually misclassified classes is observed. To get around this phenomena, the hierarchical classification model is adopted, by placing the mutually miss-classified classes together

in a separate hierarchical class. This resulted in a reduction of class size from ten to three. The hierarchical classes are Class Indoor with the airport, metro station and shopping mall as the subclasses, followed by class Outdoor with street traffic, street pedestrian, park and the public square, followed by class Vehicle with a bus, metro, tram as the sub-classes. Employing the architecture shown in the figure 7.14 , The CNN with 3-channel log-Mel spectrograms for yielded a total accuracy of 96.38 percent in classifying the sub-classes Indoor, Outdoor, and Vehicle, and sample level CNN produced an classification accuracy of 87 percent, and the DNN with smile6k features resulted in an classification accuracy of 90 percent in classifying the sub-classes.

7.8.2 Sub-class Indoor Classification

The classes airport, metro station, and shopping mall are considered to be a part of the subclass Indoor, the Log-Mel spectrograms extracted from the augmented audio samples with half-second duration are used as the input to the CNN. The architecture used to classify the hierarchical classes is used here by popping out the convolutional blocks with 256 neurons. Log-Mel Spectrograms are extracted with 128 Mel filters, the window size of 2048 samples with overlapping of 512 samples. The final result for an audio sample in the DCASE validation dataset is calculated by taking the mean of the intermediate predictions obtained from the augmented samples. This experiment resulted in the accuracy of 77.6 percent.

7.8.3 Sub-class Outdoor Classification

The classes street pedestrian, street traffic, public square, and park are considered to be the part of the subclass Outdoor. The Log-Mel spectrograms extracted with 128 Mel filters, the window size of 2048 samples and overlapping of 512 samples for the augmented data

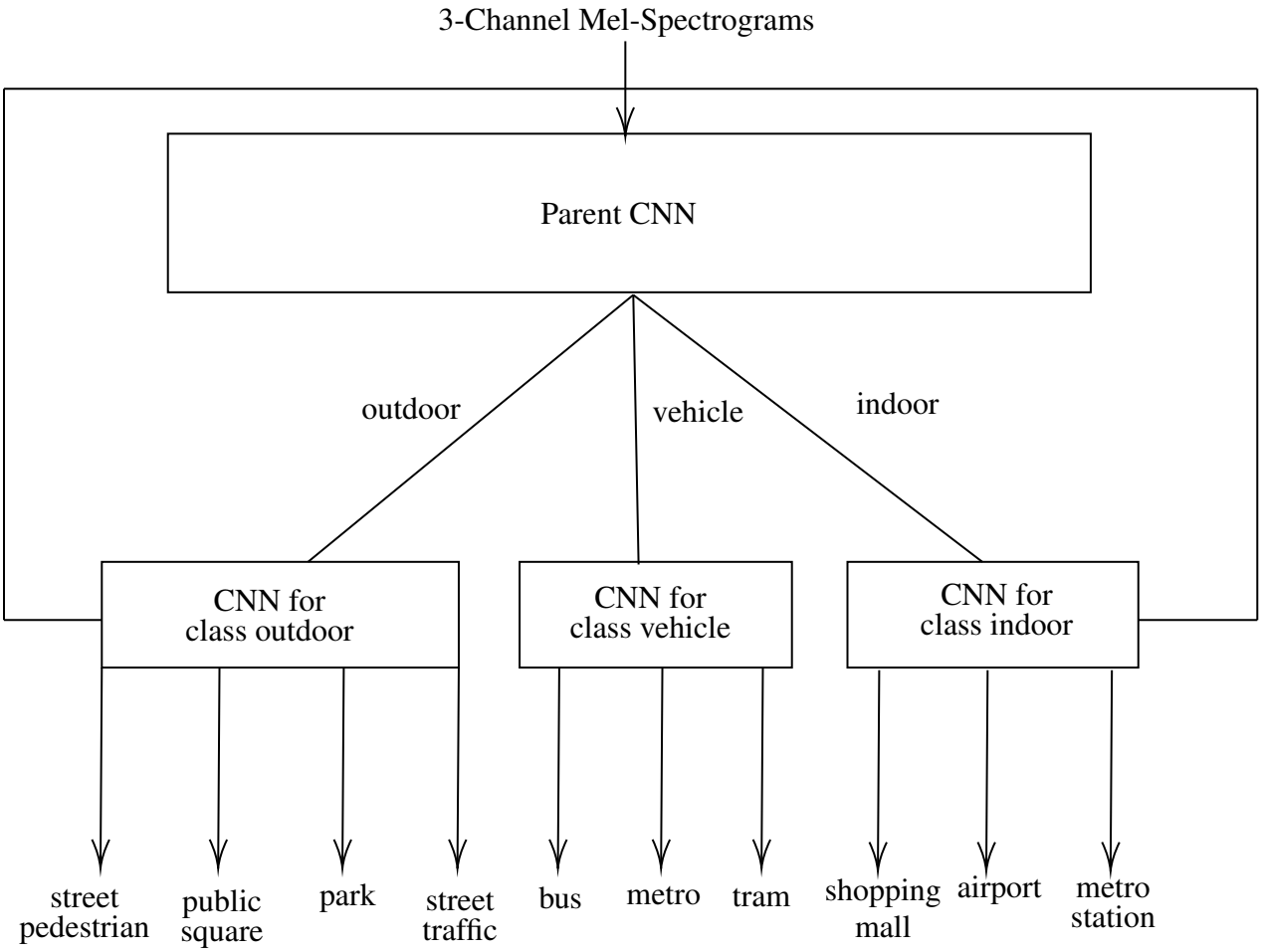
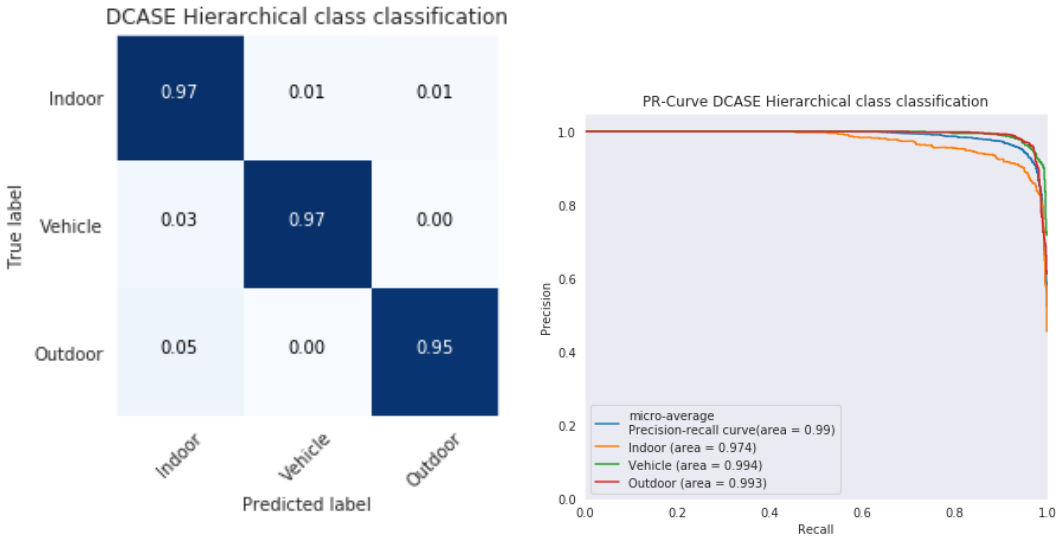


Figure 7.14: Hierarchical CNN for Acoustic Scene Classification

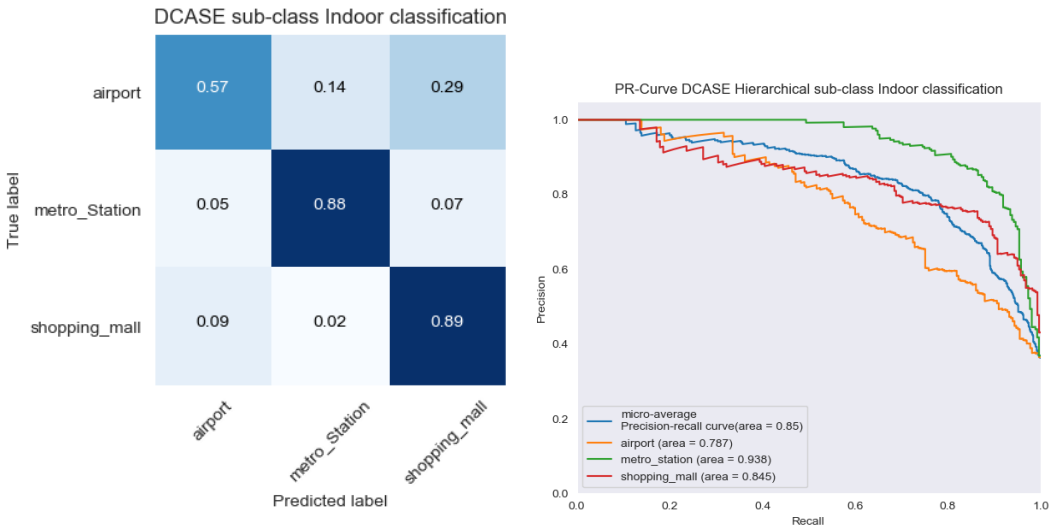


(a) Confusion Matrix (b) PR-Curve

Figure 7.15: CNN for Hierarchical class classification

Classes	Precision	Recall	F1-score	Support
Indoor	0.92	0.97	0.95	803
Vehicle	0.98	0.97	0.9	764
Outdoor	0.99	0.95	0.97	951
avg/total	0.97	0.96	0.96	2518

Table 7.8: Classification report for Hierarchical CNN



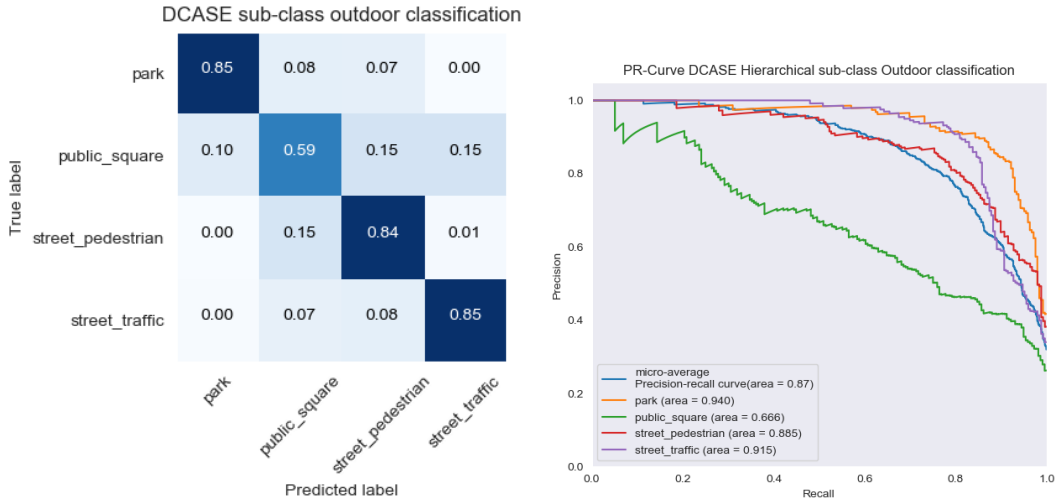
(a) Confusion Matrix (b) PR-Curve

Figure 7.16: CNN for Hierarchical class classification Sub-class Indoor

Classes	Precision	Recall	F1-score	Support
airport	0.79	0.57	0.66	265
metro_station	0.84	0.88	0.86	259
shopping_mall	0.72	0.89	0.80	279
avg/total	0.78	0.78	0.77	803

Table 7.9: Classification report for Hierarchical CNN sub-class Indoor

samples is fed into the network. The architecture used for the sub-class Indoor is used in this experiment. The final class is calculated by taking the mean of the predictions obtained on the augmented samples. This experiment resulted in the total classification accuracy of 78.4 percent.



(a) Confusion Matrix (b) PR-Curve

Figure 7.17: CNN for Hierarchical class classification sub-class Outdoor

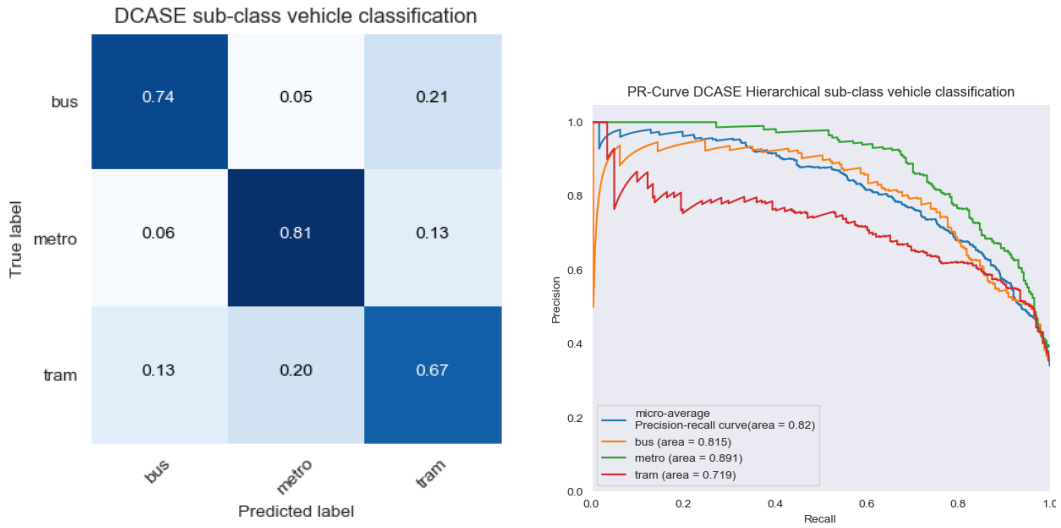
Classes	Precision	Recall	F1-score	Support
park	0.90	0.85	0.87	242
public_square	0.64	0.59	0.62	216
street_pedestrian	0.75	0.84	0.79	247
street_traffic	0.85	0.85	0.85	246
avg/total	0.79	0.78	0.78	951

Table 7.10: Classification report for Hierarchical CNN sub-class Outdoor

7.8.4 Sub-class vehicle Classification

The classes bus, metro, and tram are considered to be the part of the subclass vehicle, the Log-Mel spectrograms extracted with 128 Mel filters, the window size of 2048 samples and overlapping of 512 samples for the samples in the augmented data set. The network

is trained for 200 epochs with 1024 samples per batch. The final prediction of the sample in the DCASE validation dataset is obtained by calculating the mean of the intermediate predictions derived from the samples in the augmented DCASE dataset. This experiment resulted in the total classification accuracy of 73.7 percent.



(a) Confusion Matrix (b) PR-Curve

Figure 7.18: CNN for Hierarchical class classification sub-class Vehicle

Classes	Precision	Recall	F1-score	Support
bus	0.78	0.74	0.76	242
metro	0.77	0.79	0.62	261
tram	0.67	0.67	0.67	261
avg/total	0.74	0.74	0.74	764

Table 7.11: Classification report for Hierarchical CNN sub-class Vehicle

The final classification accuracy for DCASE dataset with hierarchical classification scheme is 74.5 percent.

8 Results, Discussion and Future Work

This chapter begins with a detailed overview of the results obtained from the experiments performed in chapter 4, chapter 6 and chapter 7. Followed by a discussion on the comparison of the results obtained to the objective of this thesis, and parameters considered in the selection of the best performing classifier. The last section of this chapter discusses the directions and scope of future work.

8.1 Results

8.1.1 Multiple Drone Detection

The value of Area Under Curve(AUC) of PR-curve is used to compare the performance of the classifiers. Higher the value of AUC, better the classifier. For each experiment, the AUC of PR-curve is reported by aggregating the values over 15 experimental trails to measure the average performance. The depth of the CNN models used for the original drone detection dataset is limited to 5, and DNN models to 2. With ten time increase in the size of augmented dataset compared to original, two-sets of results were reported on augmented drone detection dataset, with the first set of results on the exact architecture used with the original dataset, followed by using the deeper architectures for better performance. For each of the 12 feature extraction schemes employed, figure 8.1(a) showed the comparison of the performance of the classifiers with original and augmented Drone detection dataset

Feature Extraction Scheme	Model Architecture	D1	D2	D3
SMILE988	DNN	0.88	0.72	0.66
SMILE200	DNN	0.98	0.74	0.54
Spectrograms	CNN	0.66	0.94	0.57
Log-Spectrograms	CNN	0.52	0.96	0.68
Mel-Spectrograms with 128 Mel's	CNN	0.67	0.85	0.51
Log-Mel Spectrograms with 40 Mel's	CNN	0.70	0.95	0.62
Log-Mel Spectrograms with 60 Mel's	CNN	0.72	0.95	0.68
Log-Mel Spectrograms with 80 Mel's	CNN	0.69	0.95	0.67
Log-Mel Spectrograms with 128 Mel's	CNN	0.64	0.95	0.59
Log-Mel Spectrograms with 200 Mel's	CNN	0.72	0.95	0.54
Harmonic-Percussive Source Separation	CNN	0.79	0.83	0.45
Raw Audio Waveforms	CNN	0.83	No Learning	No Learning

D1: Micro-Averaged AUC(PR-Curve) for Drone Detection Dataset
D2: Micro-Averaged AUC(PR-Curve) for Augmented Drone Detection Dataset with Deeper Architectures
D3: Micro-Averaged AUC(PR-Curve) for Augmented Drone Detection Dataset.

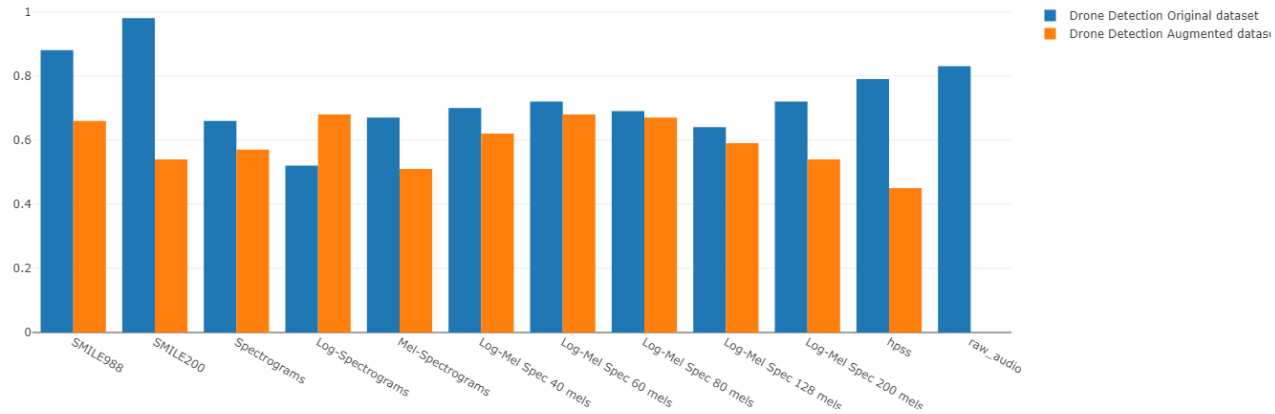
Table 8.1: Multiple Drone Detection Results

with the same architecture for the classifier. For the results shown in figure 8.1(b), deeper architectures are employed on augmented drone detection dataset.

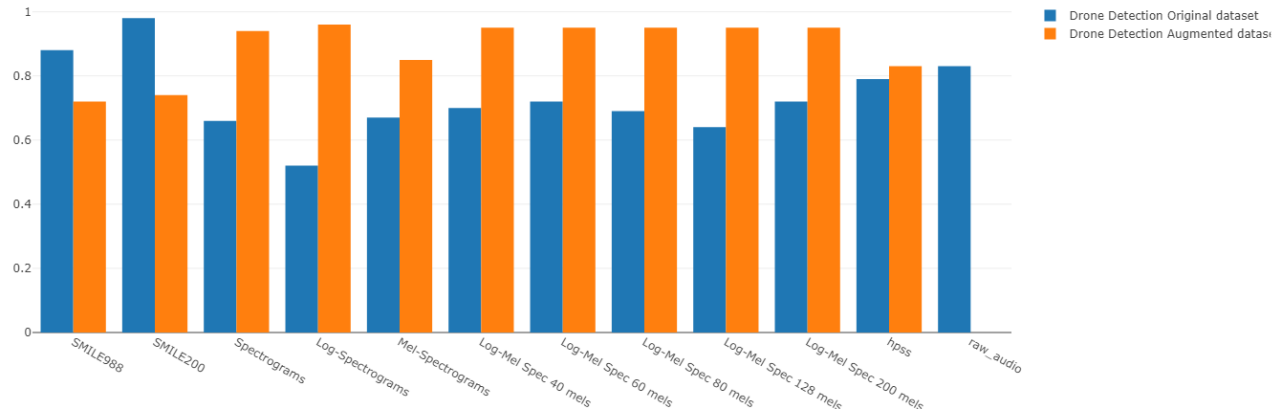
On original Drone detection dataset DNN with SMILE200 features has resulted in best AUC of 0.88 followed by CNN with raw audio waveforms with the performance of 0.83. When the same architectures are employed on augmented drone detection dataset, CNN with Log-Mel spectrograms with 128 Mel filters resulted in best performing model with AUC of 0.68. However, with the deeper architectures for Drone detection dataset resulted in AUC of 0.95 for CNN with Log-Mel Spectrograms.

8.1.2 DCASE

The value of the classification accuracy is used to compare the performance of the classifiers. Also, since the problem belongs to Kaggle competition, only the best performance of the models is used for comparison, not the average performance. Due to the limita-



(a) AUC of PR-Curve Comparision for Experiments on Original and Augmented Drone Datasets on same Architecture



(b) AUC of PR-Curve Comparision for Experiments on Original and Augmented Drone Datasets on Different Architecture

Figure 8.1: Multiple Drone Detection Results

tions in the size of the data, extending the baseline model with better architectures has not resulted in any significant performance increment. Alternatively, the large feature extraction schemes, data augmentation, and the hierarchical classification models are chosen. The smile6k feature with DNN has resulted in +10 percent over the baseline model, and smile988 with DNN resulted in +5 accuracy over the baseline model. The data augmentation scheme on the extended baseline CNN model has resulted in +9 percent accuracy over the baseline model, and the raw audio data with sample level CNN and data augmentation resulted in +1.5 percent accuracy over the baseline model, and the hierarchical classification scheme had resulted in classification accuracy of +15 percent over the baseline model.

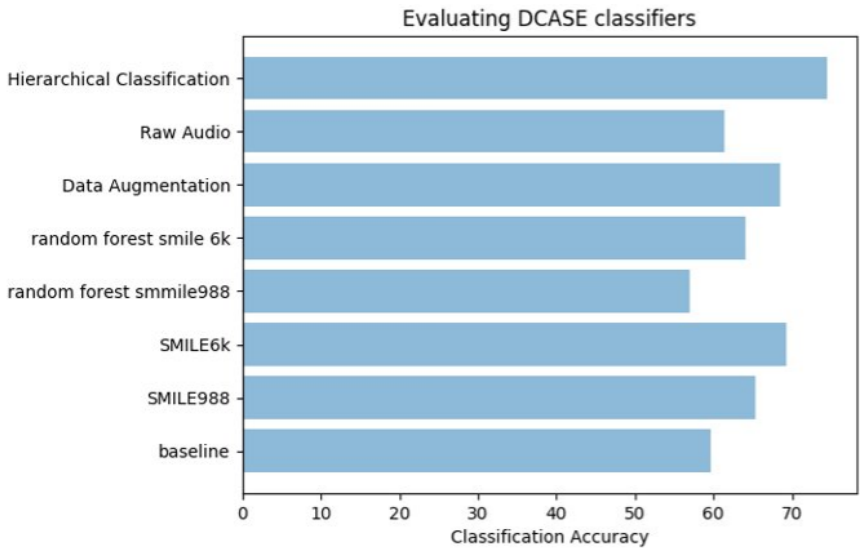


Figure 8.2: DCASE results

8.2 Discussion

8.2.1 Multiple Drone Detection

Large-scale extraction schemes with DNN and raw audio waveforms with CNN have resulted in a better performance with limited availability of data. Since, the original dataset is

collected in an isolated environment with single scene happening in the scene, the spectro-temporal domain representations are sparse. The limited data didn't allow the use of deeper architectures. Also, in spectro-temporal domain the drone audio is observed to have a significant portion of harmonic content with horizontally concentrated spectral bands at discrete frequencies, and filtering the harmonic and percussive content into two spectrogram channels resulted in better performance compared to default 3-channels spectrogram variants used in the scope of this thesis work. Generative Adversarial Networks, after 500 epochs of training started to generate perceptually hearable drone audio. On corresponding features, the performance of the classifiers, which resulted in the best performance on original dataset worked poorly with augmented data. As said earlier the deep learning algorithms are hungry for data and shown to prove the fact continuously by yielding state of the art results with the tasks, which contains millions of data samples. In our scope, this can be shown, by looking into the performance of the deep Convolutional Neural Networks with augmented drone detection audio. The raw waveform representation did not work for augmented audio, due to the noise generated by overlapping audio generated from two different distributions. Increase in the Mel filters didn't affect the performance of the classifier. Log-Mel spectrograms with 128 Mel filters is chosen as the best classifier by accounting for both the inference time and AUC of PR.

8.2.2 DCASE

For the problem of audio scene classification, the results attained can answer the questions posed in section 1.2.3 of this document. The performance of the DNN with large-scale feature extraction schemes yielded slightly better performance when compared to CNN, ignoring the ensemble models. The algorithmic techniques like employing the Exponential Linear Unit as Activation Unit(ELU) compared to Rectified Linear Unit(ReLU) coupled with RmsProp optimization algorithm resulted in faster convergence. Improving the depth of the baseline system did not increase the generalization performance over the baseline.

The VGG16 architecture, when employed with no modification, resulted in a decreased performance compared to the baseline system. However, the techniques like hierarchical classification and data augmentation by splitting each audio sample into short segments and averaging the mean predictions resulted in the increase in performance compared to the baseline system. The Log-Melspectrograms with 128 Mel filters yielded the maximum performance. Despite its performance on generating the audio data for the tasks of Automatic Speech Recognition and music generation in both time and spectro-temporal domain, the Generative Adversarial Networks have shown very poor performance on generating the acoustic scene classification data. The Convolutional Neural Networks with the Raw audio waveforms on this task yielded and classification accuracy of 61.3 percent which is 0.6 percent better than the baseline system. However, the number of experiments performed with the CNN and the raw audio is limited. Since changing the network shown in [63] slightly resulted in significantly poor performance and also computationally expensive.

8.2.3 Hyper-Parameter Search Space and statistical significance of results

Unlike the shallow machine learning algorithms, the dimensionality of the hyper-parameter search space for deep learning algorithms is significantly high. Also, limited research is been done in finding the optimal CNN architectures for audio data, compared to the field of computer vision. While working with drone detection problem, Using the grid-search approach, the hyper-parameter search was conducted over 6 dimensions, which include, depth of the model, number of neurons in each layer, optimizer, filter, and pooling size for CNN's, and batch size. For every chosen architecture five experimental trials were performed. And, to statistically increase the significance of the results, an additional ten experimental trials were performed on the model with better average performance over five trials.

8.3 Conclusions

8.3.1 Multiple Drone Detection

The questions that this thesis seeks to answer are

1) Is it possible to build an audio inference system for detecting the presence of multiple drones in the area with inexpensive Commercial Off the Shelf Equipment(COTS)?

The results obtained with both the datasets conclude that Multiple Drone Detection with the audio analysis is possible. Also, it is possible to build a Commercial Off the Shelf Equipment with audio inference system employing either CNN with Spectrograms or DNN with SMILE988 features on augmented dataset ltiple Drone Detection. However, considering the trade-off between the inference time and performance of the system, and leveraging on the online feature extraction support provided by openSMILE, DNN with SMILE988 features is chosen.

2) Assuming it can work in the prototype form, what challenges might one face in scaling drone detection for practical use?

The major challenges in scaling it for the practical purpose include the design of acoustic sensors whose range spans over long distances, and also, as said earlier in the section 1.1.1 of this document, the drone detection is only a part of a two-part process of drone detection and follow-up action performed, hence minimizing the inference time is also challenging.

3) Could the techniques used in the drone detection system prototype beat the commercial drone detection systems in performance?

The techniques used in the drone detection system prototype haven't beat the performance of the commercial systems. However, adding the audio inference system into commercially available systems will result in significant improvement of the confidence of the whole system while making the prediction.

8.3.2 DCASE

1) How would the performance of the deep learning models working on raw data in the spectro-temporal domain, compare to the performance of the statistical and probabilistic models working with Hand engineered features of both time and frequency domain?, What is the effect of the infinitely strong prior of CNN's over its weights on the solution?

From the results obtained from experiments performed in chapter 7 of this thesis work, hand-engineered features employed on Deep Neural Networks resulted in better performance. However, with hierarchical ensemble architecture, CNN's in the spectro-temporal domain have resulted in better performance.

2) Could the field of Environmental Sound classification leverage the algorithmic advances in the field of computer vision and deep learning. Which include the effect of Batch Normalization, Exponential Linear Units(ELU), Rectified Linear Unit (ReLU), an addition of Gaussian Noise in training, and Cyclic Learning Rates(CLR).

The algorithmic techniques like Gaussian Noise, Exponential Linear Units, and Batch Normalization have improved the performance of the system. ELU activation function resulted in better performance compared to ReLU, and Cyclic Learning Rates haven't shown any improvement in performance for the DCASE problem.

3) Which of the CNN architectures yield better generalization performance and faster convergence with spectro-temporal data.

Hierarchical CNN architectures have resulted in better generalization performance and faster convergence for spectro-temporal data. It is observed that the effect of CNN architecture on the performance is minimum compared to the use of data augmentation and ensemble models.

4) Which of the variants of spectrograms can yield better performance with Convolutional Neural Networks?

The three-channel Log-Melspectrograms with 128 Mel Filters and hop length of 512 have resulted in better performance among other spectrogram variants.

5) What is the performance of Convolutional Neural Networks on raw audio waveforms in time-domain?

The performance of the CNN with Raw-audio waveforms is promising. However, the spectro-temporal representation resulted in better performance.

Also as stated earlier, for the classification of acoustic scenes and sounds, the classifier needs to first identify all the events happening in a scene, followed by making a final prediction by computing the relationships between the events happening in the scene. Though the task is hard, since the classifier has to learn to identify the long-term dependencies and the relation between the features computed over various frames in audio, there still exists a significant scope of improvement for acoustic scene classification. The current state of the art classification accuracy on DCASE 2018 dataset is 79 percent, achieved with an ensemble of various models with different feature sets. However, the direct applications of the acoustic scene classifier in real-world tasks are limited, and there exist numerous applications like context-aware robots, and audio surveillance, which employs ASC classifier for intermediate state representation. The inference time needs to be minimum, and for this reason, the DNN with SMILE6k features is chosen as the best classifier compared to the hierarchical acoustic scene classification

8.3.3 Future Work

Up until the year 2016, audio analysis with neural networks involved an intense amount of domain knowledge and heavy feature engineering, which ended up adding significant amount human bias into the classifier. The exponential growth in the field of computer vision in the current decade resulted in several algorithmic techniques. These advances are leveraged to decrease the convergence time and increase the performance accuracy with spectro-temporal features of audio. However, there's still a lot to catch on, and the recent works [70, 63, 69, 71] have shown promising results in that way using the raw audio waveforms as input to the CNN. In future, the author wants to involve in experimenting on to find

the CNN architectures and algorithmic techniques, which increases the classifier's performance on the tasks of audio pattern recognition, sound event detection, automatic speech recognition, and audio scene classification with raw audio as the input feature vector. And, work with the generative adversarial networks for environmental or acoustic scene sounds.

Bibliography

[1] Federal register notice: Unmanned aircraft operations in the national airspace system. https://www.faa.gov/uas/media/frnotice_uas.pdf. Accessed: 2018-09-28.

[2] Milan Erdelj, Enrico Natalizio, Kaushik R Chowdhury, and Ian F Akyildiz. Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, (1): 24–32, 2017.

[3] Dane Bamburry. Drones: Designed for product delivery. *Design Management Review*, 26(1):40–48, 2015.

[4] Cornelius A Thiels, Johnathon M Aho, Scott P Zietlow, and Donald H Jenkins. Use of unmanned aerial vehicles for medical product transport. *Air medical journal*, 34 (2):104–108, 2015.

[5] Ludovic Apvrille, Tullio Tanzi, and Jean-Luc Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. In *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pages 1–4. IEEE, 2014.

[6] Shripad Gade, Aditya A Paranjape, and Soon-Jo Chung. Herding a flock of birds approaching an airport using an unmanned aerial vehicle. In *AIAA Guidance, Navigation, and Control Conference*, page 1540, 2015.

[7] Nicola Ceccarelli, John J Enright, Emilio Frazzoli, Steven J Rasmussen, and Corey J Schumacher. Micro uav path planning for reconnaissance in wind. In *American Control Conference, 2007. ACC'07*, pages 5310–5315. IEEE, 2007.

[8] Katia Moskvitch. Take off: are drones the future of farming? *Engineering & Technology*, 10(7-8):62–66, 2015.

[9] Drone crash near white house. https://www.cbsnews.com/news/drone-crash-lands-white-house/, . Accessed: 2018-09-28.

[10] Geo spatial environment online lock. https://www.dji.com/flysafe. Accessed: 2018-09-28.

[11] Drone detection with acoustic sensors. www.aerodefensetech.com/component/content/article/adt/features/articles/33023. Accessed: 2018-09-28.

[12] Drone detection with doppler-based radar. anti-drone.eu/products/acoustic-sensors/. Accessed: 2018-09-28.

[13] Dronewatcher layered drone surveillance and interdiction. https://detect-inc.com/drone-detection-defense-systems/. Accessed: 2018-09-28.

[14] Sungho Jeon, Jong-Woo Shin, Young-Jun Lee, Woong-Hee Kim, YoungHyoun Kwon, and Hae-Yong Yang. Empirical study of drone sound detection in real-life environment with deep neural networks. *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1858–1862, 2017.

[15] Andrea Bernardini, Federica Mangiatordi, Emiliano Pallotti, and Licia Capodiferro. Drone detection by acoustic signature identification. *Electronic Imaging*, 2017:60–64, 01 2017. doi: 10.2352/ISSN.2470-1173.2017.10.IMAWM-168.

[16] Louise Hauzenberger and Emma Holmberg Ohlsson. Drone Detection using Audio Analysis. Master's thesis, Lund University, Sweden, 2015.

[17] Jurgen T Geiger, Bjorn Schuller, and Gerhard Rigoll. Large-scale audio feature extraction and svm for acoustic scene classification. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, pages 1–4. IEEE, 2013.

[18] Jort F Gemmeke, Lode Vuegen, Peter Karsmakers, Bart Vanrumste, et al. An exemplar-based nmf approach to audio event detection. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, pages 1–4. IEEE, 2013.

[19] Onur Dikmen and Annamaria Mesaros. Sound event detection using non-negative dictionaries learned from annotated overlapping events. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, pages 1–4. IEEE, 2013.

[20] Juhan Nam, Ziwon Hyung, and Kyogu Lee. Acoustic scene classification using sparse feature learning and selective max-pooling by event detection. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[21] Axel Plinge, Rene Grzeszick, and Gernot A Fink. A bag-of-features approach to acoustic event detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3704–3708. IEEE, 2014.

[22] Lode Vuegen, BVD Broeck, Peter Karsmakers, JF Gemmeke, Bart Vanrumste, and HV Hamme. An mfcc-gmm approach for event detection and classification. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–3, 2013.

[23] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. CP-JKU submissions for DCASE-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks. Technical report, DCASE2016 Challenge, September 2016.

[24] Yoonchang Han and Kyogu Lee. Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification. Technical report, DCASE2016 Challenge, September 2016.

[25] Thomas Lidy and Alexander Schindler. CQT-based convolutional neural networks for audio scene classification and domestic audio tagging. Technical report, DCASE2016 Challenge, September 2016.

[26] Yoonchang Han and Jeongsoo Park. Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification. Technical report, DCASE2017 Challenge, September 2017.

[27] Zheng Weiping, Yi Jiantao, Xing Xiaotao, Liu Xiangtao, and Peng Shaohu. Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion. Technical report, DCASE2017 Challenge, September 2017.

[28] Bernhard Lehner, Hamid Eghbal-Zadeh, Matthias Dorfer, Filip Korzeniowski, Khaled Koutini, and Gerhard Widmer. Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task. Technical report, DCASE2017 Challenge, September 2017.

[29] Abdul J Jerri. The shannon sampling theorem—its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[30] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[31] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[32] J Allen. Applications of the short time fourier transform to speech processing and spectral analysis. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, pages 1012–1015. IEEE, 1982.

[33] David L Donoho and Philip B Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

[34] Derry Fitzgerald. Harmonic/percussive separation using median filtering. 2010.

[35] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE transactions on audio, speech, and language processing*, 15(3):1066–1074, 2007.

[36] Francisco Jesus Canadas-Quesada, Pedro Vera-Candeas, Nicolas Ruiz-Reyes, Julio Carabias-Orti, and Pablo Cabanas-Molero. Percussive/harmonic sound separation by non-negative matrix factorization with smoothness/sparseness constraints. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014(1):26, 2014.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning

for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

[45] Zhifei Zhang. Derivation of backpropagation in convolutional neural network (cnn). 2016.

[46] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[47] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. URL http://arxiv.org/abs/1506.01186.

[48] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

[49] Richard S. Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1986.

[50] Neural networks for machine learning: Overview of mini-batch gradient descent. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Accessed: 2019-01-04.

[51] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

[52] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[53] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[54] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[55] Keunwoo Choi, Deokjin Joo, and Juho Kim. Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras. *CoRR*, abs/1706.05781, 2017. URL http://arxiv.org/abs/1706.05781.

[56] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[57] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM, 2010.

[58] Configuration file employed for smile988 feature extraction. https://github.com/naxingyu/opensmile/blob/master/config/emobase.conf, . Accessed: 2018-09-28.

[59] Configuration file employed for smile6k feature extraction. https://github.com/naxingyu/opensmile/blob/master/config/emo_large.conf, . Accessed: 2018-09-28.

[60] Florian Eyben. *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*. 01 2016. doi: 10.1007/978-3-319-27299-3.

[61] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45:427–437, 2009.

[62] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143874. URL http://doi.acm.org/10.1145/1143844.1143874.

[63] Taejun Kim, Jongpil Lee, and Juhan Nam. Sample-level cnn architectures for music auto-tagging using raw waveforms. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr 2018. doi: 10.1109/icassp.2018.8462046. URL http://dx.doi.org/10.1109/ICASSP.2018.8462046.

[64] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[65] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016.

[66] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing audio with generative adversarial networks. *arXiv:1802.04208*, 2018.

[67] Audio sample for single drone in the scene class generated by wavegan. https://drive.google.com/file/d/1VTN_jr7mh8WA4i6nvKe0scHHldZVSlEA/view?usp=sharing, . Accessed: 2018-09-28.

[68] Dai Wei, Juncheng Li, Phuong Pham, Samarjit Das, and Shuhui Qu. Acoustic scene recognition with deep neural networks (dcase challenge 2016). 2016.

[69] Jongpil Lee, Taejun Kim, Jiyoung Park, and Juhan Nam. Raw waveform-based audio classification using sample-level cnn architectures. *CoRR*, abs/1712.00866, 2017.

[70] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125, 2016.

[71] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms, 2017.